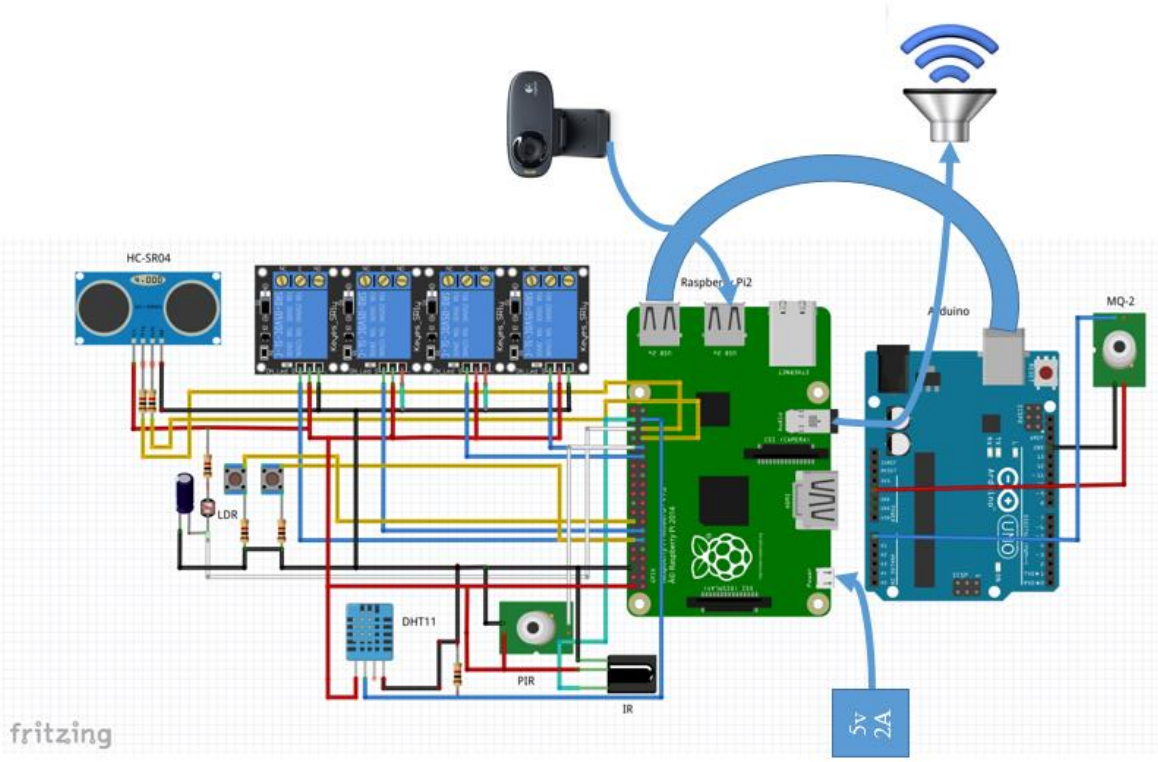
TSHWANE UNIVERSITY OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING

Course: INDUSTRIAL PROJECT IV

PROJECT REPORT:

DESIGNING A SMART HOME AUTOMATION WITH VOICE RECOGNITION USING RASPBERRY PI AND ARDUINO



Student Name: Mpho Mphego

Student No.: 205044990

Internal supervisor:

External or co-supervisor:

Receiving date:

Table of Contents

Page no.

Table of Figures

Page no.

List of Tables

Page no.

Abstract

This Smart Home Automation system offers convenience and high-level control over a home environment, security and media systems from a centralised network core with an easily accessible web and mobile interface without any changes in the infrastructure as it is to be implemented in an existing home environment.
The environmental system can control the lighting, plugs and temperature of the house. The security system will constantly monitor for any intrusion through windows and doors and give the user remote control over door locks.
The media system will let user's stream music from the internet or a local source to multiple zones throughout the house.
The integration of these everyday systems will give the average home-owner the control they desire within their home.

1 Chapter 1

1.1 Introduction

As technology advances a lot of automation implementation in various fields has been introduced in order to maintain security, time and cost. In this process our homes lags behind, even though a lot of advanced equipment's are introduced each year, the use of these equipment are limited in the context of our homes. The most critical obstruction in home automation is the availability of these technologies and the cost implications involved as well as the maintenance.

Hence the idea to design and build a low cost home automation product which does not allow any sophisticated installations and home infrastructure excessive modifications. The major concern in this case is affordability, usability and security - which leads us to the design of a low cost home automation system which offers multiple control and monitoring interfaces such as the use of mobile devices and/or computers with internet connection.

The main aim of the project is to remove all limitations and obstructions in home automation by taking the correct approach. The Smart Home-Automaton System will consist of sensors to detect current status in the house such as temperature and humidity, lightning conditions and presence detection and many more will be discussed below. The system will have the ability to determine and automatically adjust temperatures or lighting depending on the user preferences.

Other features include gesture control by means of shaking mobile device or voice control to switch on/off any appliances that the user configured, a web-based graphical user interface listings current temperatures(outdoors and indoors), lighting controls and real-time video streaming(security purposes).

1.2 Problem statement

The main problem statement for the current system being developed is the ability to make life a lot easier by means of automating household equipment. A good example would be the automatic washing machine which helped in transforming washing which is the most hardest and dull domestic duties in a household to being one of the least burdensome work. Humans have gotten to the point where they prefer simplified and non-sophisticated systems to simplify their livelihood. This led to the development of the smart home automation system which offers variety of solutions such as lighting, cooling/heating systems as well as security systems, it also offers multi-room audio system with future improvements including natural language processing for voice controlled applications.

However, most current systems in the market do not conveniently integrate the aforementioned house systems in one product. Instead, the buyer must purchase various devices and integrate them to form a system, which often requires significant technical knowledge. Furthermore, most commercially available products integrate their products using a power line technology instead of wireless network. This leads to decreased network security and increased hassle for installation.

1.3 User requirement specification

In order to demonstrate the smart home automation system we will use various components such as Microcontroller, Sensors, Relays, Mobile phone or computer and most importantly the brains of the project which is an SBC (Single Board Computer

The proposed system should be low cost, modifiable or monitored remotely.

It should incorporate the below listed items.

- Uninterrupted Power Supply.
- HVAC control.
- Temperature and Humidity monitoring (outdoors and indoors).
- Multi room media player/server
- LPG Gas/Smoke monitoring
- Wi-Fi-based passive Wi-Fi presence detector and PIR
- Smart Alarm
- Smart closet
- Smart doorbell
- TV proximity sensors
- Web-based control and monitoring platform.
- Android mobile control and monitoring

1.4 Study objectives

On the completion of this project knowledge, experience and skills will be gained in the following fields:

- **Single board micro-controllers**

A deep knowledge on single board micro-controllers will be gained from this project, as a micro-controller will be used at the center of the design to interface diverse and complex devices and components. This knowledge will be initiated by an overview on different micro controller's characteristics followed by the choice of an adequate microcontroller that will suit the system requirement.

- **Power supplies and Analogue/Digital Electronics**

Since the final product will be a mixture of electronic components which need to be powered, a suitable source of power should be studied to ensure the functionality of each of these components and the autonomy of the entire system (device). Furthermore, having different electrical specifications, proper Electronics theory will be applied.

- **Computer networking**

At the completion of the project an understanding and experience in computer networking will be gained and at a comprehensive level, some protocols used in the project includes TCP/IP, UDP, SMTP and HTTP.

- **Circuit Simulation software packages**

Another skill to be acquired from this project is the one of using CAD and simulation software such as **Fritzing, Altium Designer, Proteus, LT Spice and KiCad.**

- **Programming Skills**

Programming skills is another benefit gained from this project. As stated previously, a microcontroller is at the center of the design. This microcontroller needs to be loaded with a set of instructions (code) in a certain language that can be interpreted into machine language and instructs to the microcontroller the functionality of the system. The primary programming language for the project is **Python, JAVA, C and Bash**.

- **Signal Processing**

The most important skill to be acquired at the completion of the project is signal processing as we will be reading real time electronic signals (by means of using filters and **FFT** (Fast Fourier Transform) algorithms) and interpreting them to readable information or instructions.

- **Research**

A considerable part of knowledge required for the completion of this project is not part of the syllabus of the academic program and will therefore require some research and a literature review.

1.5 Definitions

TCP - Transmission Control Protocol is a standard that defines how to establish and maintain a network conversation via which application programs can exchange data.

API - Application program interface

UDP - User Datagram Protocol is an alternative communications protocol to Transmission Control Protocol (TCP) used primarily for establishing low-latency and loss tolerating connections between applications on the Internet.

SMTP - Simple Mail Transfer Protocol is an Internet standard for electronic mail (email) transmission.

FFT - A fast Fourier transform (FFT) algorithm computes the discrete Fourier transform (DFT) of a sequence, or its inverse. Fourier analysis converts a signal from its original domain (often time or space) to a representation in the frequency domain and vice versa.

Wi-Fi - A local area wireless computer networking technology that allows electronic devices to network.

DLNA - Digital Living Network Alliance is a standard that allows various consumer electronic devices to share content with each other across a standard home network.

UPnP - Universal Plug and Play is an Internet protocol set primarily for home networks permitting devices to access the network.

IR - Infrared radiation refers to energy in the region of the electromagnetic radiation spectrum at wavelengths longer than those of visible light.

HVAC - heating, ventilating, and air conditioning is the technology of indoor and vehicular environmental comfort.

DHCP - The Dynamic Host Configuration Protocol is a standardized network protocol used on Internet Protocol (IP) networks for dynamically distributing network configuration parameters, such as IP addresses for interfaces and services.

IP - Internet Protocol is the method or protocol by which data is sent from one computer to another on the Internet.

LDR - Light Dependent Resistor or Photo resistor, is a passive electronic component, basically a resistor which has a resistance that varies depending of the light intensity.

LAN - A local area network is a network that connects computers and other devices in a relatively small area, typically a single building or a group of buildings

SMS - Short Message Service (SMS) is a text messaging service component of phone, Web, or mobile communication systems.

IoT - The Internet of Things (IoT) refers to the
ever-growing network of physical objects that feature an IP address for internet connectivity, and the communication that occurs between these objects and other Internet-enabled devices and systems.

PC - Personal Computer

GUI - Graphical User Interface

Sensor - A device that detects and responds to some type of input from the physical environment.

Push notifications - lets your application notify a user of new messages or events even when the user is not actively using your application. On Android devices, when a device receives a push notification, your application's icon and a message appear in the status bar.

RPi - A Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse.

Web-cam - Is a digital camera that's connected to a computer. It can send live pictures from wherever it's sited to another location by means of the internet.

SBC - A single board computer is a self-contained computer that only requires a power supply for operation.

ADC - Analog to Digital Convention.

API - Application program interface

1.6 Importance and benefits of the study

There are various commercially available products in the home automation industry that perform functions similar to the smart home automation system.

However, most current systems in the market do not conveniently integrate the aforementioned house systems in one product. Instead, the buyer must purchase various devices and integrate them to form a system, which often requires significant technical knowledge. Furthermore, most commercially available products integrate their products using a power line technology instead of wireless network. This leads to decreased network security and increased hassle for installation.

Why is there a need for another system of this type?

The proposed solution adds various unique features not currently found in this market that make the proposed system unique:

- The proposed system will allow historical data gathering of the system so that they can be viewed by the user to help detect any significant changes by means of logging. This can help in the diagnostics of problems by being able to see past data. Many top end building automation systems provide this feature, but the lower end home automation systems do not.
- Most of the automation systems on the market today have the ability to send alarms via e-mail or by pre-recorded voice messages. The problem with these systems is that if the system fails (loss of power, lighting strike, internet service goes down) the remote interface does not indicate a problem until the operator tries to access the system. In many cases the time when you need it most, like loss of power, is the time the system is unable to warn the owner of the problem.

  The solution to this problem is to have the application on the smart-phone test the status of the system periodically. If the system fails to respond in a present length of time the smart phone can notify the owner that there may be a potential problem. This handshake between the systems allows for a more reliable warning system than the currently available 'call out' systems.
- By choosing a wireless communication medium, we could save wiring and installation cost as we could reuse existing infrastructure with minimal modification. By developing an application that enables users to control devices from the computer instead of a dedicated

console, we could save the cost for the need of a dedicated automation console.

1.7 Budget

The rough estimation of the project is R3910.00. Refer to appendix A.1 for a detail layout of the budget.

Table 1-1 Budget of the proposed project

Components	Cost [R]
Raspberry Pi B+	850.00
Arduino Uno R3	190.00
USB Webcam	120.00
USB Wi-Fi Dongle	100.00
USB/3.5mm Speakers	50.00
Various Sensors[1]	800.00
5V 2A Power supply	150.00
2x Relay modules	250.00
Enclosure	200.00
Other electronic components (approximately)	200.00
Total component cost:	R3910.00

[1] Various sensors used include PIR, GAS (MQ2), IR, Ultrasonic (HC-SR04), Temp/Humidity (DHT11) sensors(s).

1.8 Conclusion

Chapter 1 discusses and explains the various aspects of the project proposed and the importance therein as well as the estimated cost implications. Most importantly the benefits of the project in terms of everyday life and student gains in terms of skills and knowledge of how things work as well as the user requirements.
More details will be explained in the following chapters.

Chapter 2 discusses and explains the user 0requirement and specification of the proposed design of the project.
Chapter 3 explains the development and practical implementation as proposed in chapter 2 and also how it will be tested.
Chapter 4 shows and discusses, results of the working model or product.
Chapter 5 evaluates the success of this project and gives some recommendations for improvements and Conclusions.

2 Chapter 2

2.1 Introduction

As the world of technology advances a lot of automation has been introduced in various fields in order to maintain with the time, and there has always been a need to automate applications in a home such as dish washing, laundry washing machines and other automated appliances which at the end of the day improves human lives. In this chapter we will discuss the proposed projects, requirements and specifications to transform an ordinary home to an automated home.

2.2 Literature review

As per my research, there exist a lot of home automation systems across the board using a single board computer such as the Raspberry Pi, with its processing power and connectivity a lot of project have been researched and done in the past. But nothing as compared to the project that is being discussed which is the smart home automation system based on the raspberry pi and Arduino single board computers. Each system that will be discussed below has its own unique features.
The following project have been research and compared with the one in question.
In this section the different types of home automation are discussed:

- Android based home automation using raspberry pi - Shaiju Paul, Ashlin Antony and Aswathy B
- HomeAutomation – Pratik Gadtaula
- Qwikswitch – light warehouse

Even though there are many available solutions of home automation, the systems are however limited. The current systems are implemented with a number of hardware, in most cases installation and maintenance can be a difficult task on its own. They also impose huge installation costs on the user or consumer.

2.2.1 Android Based Home Automation Using Raspberry Pi

The home automation system is working with very popular android phones. It is having mainly three components; the android enabled user device, a Wi-Fi router having a good scalable range, and a raspberry pi board .Here the users have provision to control the home appliances through android enabled device. This will improve the system popularity since there is no need for a wired connection, internet etc. The instructions from the user will be transmitted through the Wi-Fi network .The raspberry pi board is configured according to the home system and it will enable the relay circuit as per user request. The relay circuit can control the home appliances also. We can add appliances to the system also can add additional security features. The main objectives of the proposed system is to design and to implement a cheap and open source home automation system that is capable of controlling and automating most of the house appliances through an android device.[1]

Advantages of the android based home automation system.

- The new system must provide the following features
- It allows more flexibility through android device.
- It allows a good range of scalability.
- It provides security and authentication.
- Additional vendors can be easily added.

Figure 2-1 Android based Home automation using Raspberry Pi

2.2.2 HomeAutomation

The product covers the area of monitoring and controlling appliances in home as per user's configuration and control. As the automation is performed on Raspberry Pi device along with Arduino board, it combines the overall benefits from both devices and thus useful in implementing our tasks. It primarily focus on safety and then other facilities extended along with it. Services like knowing temperature reading, lights On/Off condition, fan On/Off and other services are featured in this Home Automation. The Alarm system is also major part in Home Automation which secure the home and update user with right information in right time to avoid accident and loss. The controlling section is great importance in Home Automation. User will have automatic settings to control the appliances. Further, this service is good and one of the reliable way to encapsulate home from internal and external danger. People in job or outside home can work freely and smartly having control to their home. They can just sit and login browser and see what is going on in their home in just a second and feel that their home is with them all time. Home Automation is truly one of the needs in today's world. People rely and feel safe and warmth in their home with their family. Home Automation brings closer and safer to them. [2]

2.2.3 Qwik Switch

The Qwikswitch from The Lighting Warehouse provides affordable wireless lighting control that is easy to install and operate. It presents a number of various benefits, including [3]

- It will save you time, labor and wire.
- It is very useful when renovating or adding a switch to a room, as it allows you to move switches as your needs change and eliminate unnecessary wiring.
- The Qwikswitch switch plates are easy to install with double sided tape or screws.
- The Qwikswitch makes multi way switching or dimming a quick and simple process.
- A remote control means you never need to enter your home in the dark - activate interior or exterior lights from the safety of your car.

Figure 2-2 QwikSwitch

The Qwikswitch however offers very limited security and minimal control, as compared to the Smart home automation.

2.3 Proposed practical design or strategy

This section provides an overview of the system and steps in the developments process. The system interfaces with external aspects. Each of the external aspects is described in the text below. Figure below shows the interactions among the various interfaces of the system as per the designer.

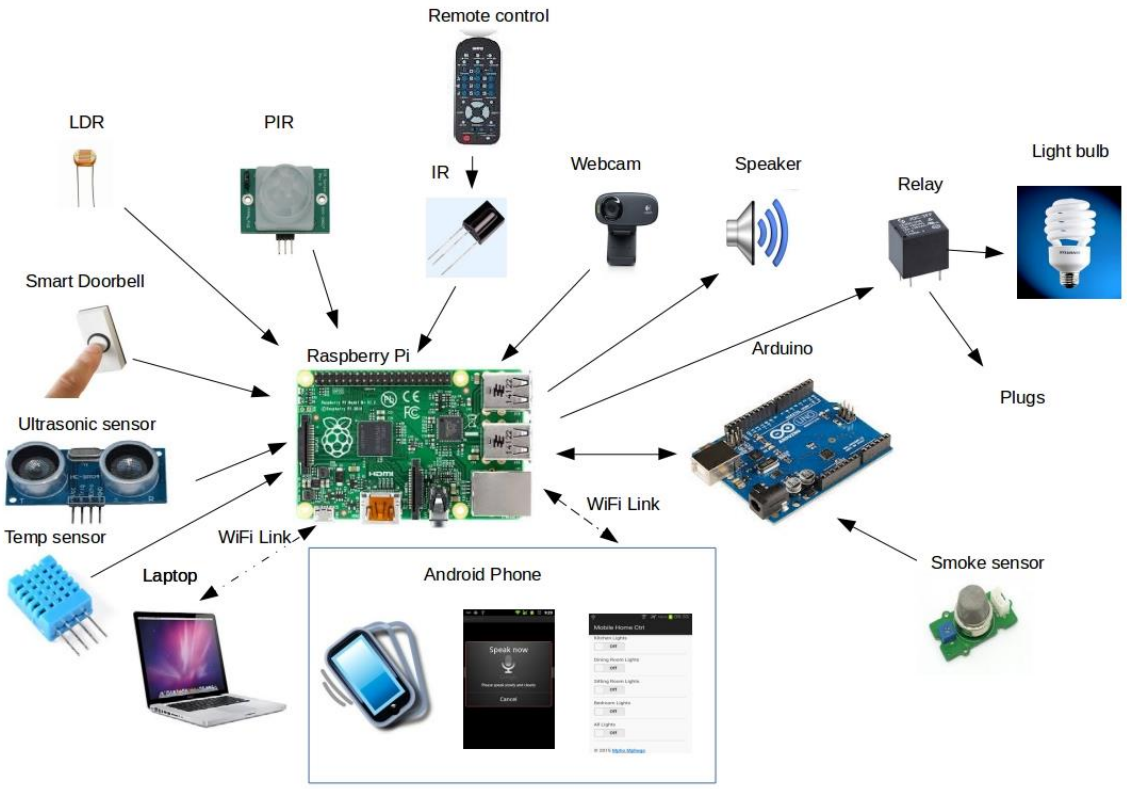


Figure 2-3 Diagram of proposed system

The system can be modified or monitored remotely.

This kind of system presents many advantages compared to others on the same market.

- Uninterrupted Power Supply in the case of power outages.
- HVAC control using IR and temperature and humidity monitoring (outdoors and indoors).
- Multi room Audio player (DLNA/UPnP and web-based).
- LPG Gas/Smoke detection with E-Mail, push notification and siren notification in the case of leaks.
- Wi-Fi-based passive Wi-Fi presence detector, which can be enabled to switch on/off certain appliances/lights depending on the time.

- Smart Alarm which wakes the home-
  owner up and then reads out the current weather and the day's forecast, as well as the current news while it opens up the blinds. It switches the coffee maker on such that as soon as the home-owner is ready to leave the house, he/she can enjoy a freshly brewed cup of coffee.
- Smart closet which notifies home-owner to carry a jacket or umbrella before they leave the house depending on the weather.
- Smart doorbell which sends an SMS/E-mail and Push notification as well as takes a picture of the visitor and sends to home-owner.
- TV proximity sensors, to avoid kid's straining their eyes by standing close to the TV.
- Web-based control and monitoring platform.
- Android mobile based control and monitoring including Voice recognition and gesture control, by means of shaking the mobile device certain light will be switch on/off in the house.

2.4 Product specifications or requirements

Raspberry Pi B+

The Raspberry Pi is a low cost, credit-card sized computer that is running Debian Linux operating systems it is the major component of the system also called as single board computer.

This computer is the brains of the project all codes will be written in Python and C programming language.

Figure 2-4 Raspberry Pi B+

Arduino Uno

Arduino Uno is a microcontroller board based on the Atmel ATmega328P. It is responsible for all Analog to Digital related issues as the Raspberry Pi does not come with an ADC chip on it.

It also has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs which will be used for connecting to the outside world, it comes with a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. This microcontroller codes will be written in C programming language.

Figure 2-5 Arduino Uno R3

Relays

A Relay is an electrically controlled switch used for switching a power circuit, similar to a relay except with higher current ratings. A relay is controlled by a circuit which has a much lower power level than the switched circuit. It will be responsible for switching lights, plugs and anything else requiring more than 12V to switch on/off.

LDR

A photo resistor or light dependent resistor is a resistor whose resistance decreases with increasing incident light intensity; in other words, it exhibits photo conductivity.

Speaker

A loudspeaker (or "speaker") is an electro acoustic transducer that produces sound in response to an electrical audio signal input. This is used for giving voice information. Such as weather and presence notifications.

Webcam

A webcam is a video camera that feeds its images in real time to a computer or computer network, often via USB, Ethernet, or Wi-Fi. It is mainly used in our project for security surveillance and computer vision.

IR and Remote

An infrared sensor is an electronic device that emits and/or detects infrared radiation in order to sense some aspect of its surroundings. Infrared sensors can measure the heat of an object, as well as detect motion. Many of these types of sensors only measure infrared radiation, rather than emitting it, and thus are known as passive infrared (PIR) sensors.

Smoke Sensor

The MQ series of gas sensors use a small heater inside with an electro-chemical sensor. They are sensitive for a range of gasses and are used indoors at room temperature. The output is an analog signal and can be read with an analog input of the Arduino. The MQ-2 Gas Sensor module is useful for gas leakage detecting in home and industry. It can detect LPG, I-butane, propane, methane, alcohol, hydrogen and smoke.

PIR

A passive infrared sensor (PIR sensor) is an electronic sensor that measures infrared (IR) light radiating from objects in its field of view. They are most often used in PIR-based motion detectors. It is as a motion detector and presence detector.

Smart doorbell

A smart Alarm which wakes the home-owner up and then reads out the current weather and the day's forecast, as well as the current news while it opens up the blinds. It switches the coffee maker on such that as soon as the home-owner is ready to leave the house, he/she can enjoy a freshly brewed cup of coffee.

Ultrasonic sensor

An ultrasonic sensor will be used for kids distance detection from the TV to mitigate any form of eye straining and disturbance. If a kid stands close to the TV it sets an alarm notify them to move away from the TV else the TV is switched off.

Temperature sensor

A digital sensor senses temperature and humidity of the surrounding and converts it into an electrical signal that can be read by other electronic instrument.

Laptop

A Web-based control and monitoring platform accessible on a laptop connected to the local network.

Android-based Phone

Android mobile based control and monitoring including Voice recognition and gesture control, by means of shaking the mobile device certain light will be switch on/off in the house.

2.5 Conclusion

According to this review of the current systems and the proposed system, one can easily point out why the proposed system is unique and it is a low cost system. The proposed solution adds various unique features not currently found in the market this makes the proposed system unique, See Importance and benefits of the study.

3 Chapter 3

3.1 Introduction

This chapter explains the development and practical implementation in detail as proposed in chapter 2. We will discuss each feature of the proposed project in detail. We will go through the features in alphabetical order.

3.2 Design or development of product / strategy

3.2.1 Android Mobile Control

The Android mobile application was written in Java under Android Studio.
Android Studio is the official IDE for Android app development, based on IntelliJ IDEA. On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance your productivity when building Android apps.

The mobile application was designed with simplicity in-mind, it uses the Android System Webview which accesses Google's chrome with minimal features, and the application on start-up automatically connects to the Raspberry Pi Apache webserver considering that the mobile phone is connected to the same network as the Raspberry Pi.

What is Android's Webview?
Android's Webview, as described by Google, is a "system component powered by Chrome that allows Android apps to display web content." In other words, Webview allows 3rd party apps to show content in an in-app browser or in an app screen that pulls from the web. It's pretty important, and has only recently (with Lollipop) been decoupled as a stand-alone system component that can be updated as Google sees fit. And that's important, because it allows Google to push security fixes and other enhancements without the need to push an entire system update.

The code below is the main java and xml code for the application.

Source code:

Main_Activity.java

```
//__author__ = "Mpho Mphego"
//__description__ = "Android WebView App controlling Raspberry Pi running Apache2"
//__version__ = "Revision: 1.0 "
//__date__ = "Date: 2015/01/17 22:23 "
//__url__ = "mpho112.wordpress.com"
//__copyright__ = "Copyright (c) 2015 Mpho Mphego"
//__license__ = "Java"

package com.wordpress.mpho112.mobilehomectrl;
import android.os.Bundle;
import android.support.v7.app.ActionBarActivity;
import android.view.Menu;
import android.view.MenuItem;
import android.webkit.WebSettings;
import android.webkit.WebView;

public class MainActivity extends ActionBarActivity {
   private WebView mWebView;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      mWebView = (WebView) findViewById(R.id.activity_main_webview);
      // Enable Javascript
      WebSettings webSettings = mWebView.getSettings();
      webSettings.setJavaScriptEnabled(true);
      mWebView.loadUrl("http://raspberrypi.local");
   }
   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      // Inflate the menu; this adds items to the action bar if it is present.
      getMenuInflater().inflate(R.menu.menu_main, menu);
      return true;
   }
   @Override
   public boolean onOptionsItemSelected(MenuItem item) {
      // Handle action bar item clicks here. The action bar will
      // automatically handle clicks on the Home/Up button, so long
      // as you specify a parent activity in AndroidManifest.xml.
      int id = item.getItemId();

      //noinspection SimplifiableIfStatement
      if (id == R.id.action_settings) {
         return true;
      }
      return super.onOptionsItemSelected(item);
   }
}
```

Activity_main.xml

```
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:id="@+id/container"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">
tools:ignore="MergeRootFrame">

<WebView
android:id="@+id/activity_main_webview"
android:layout_width="match_parent"
android:layout_height="match_parent" />
</FrameLayout>
```

The Figure 3-1 Android Control below display's how the mobile control interface currently looks like - with its simplistic controls. It features only the lighting controls. More development shall follow, for instance adding an indoor temperature monitoring gauge

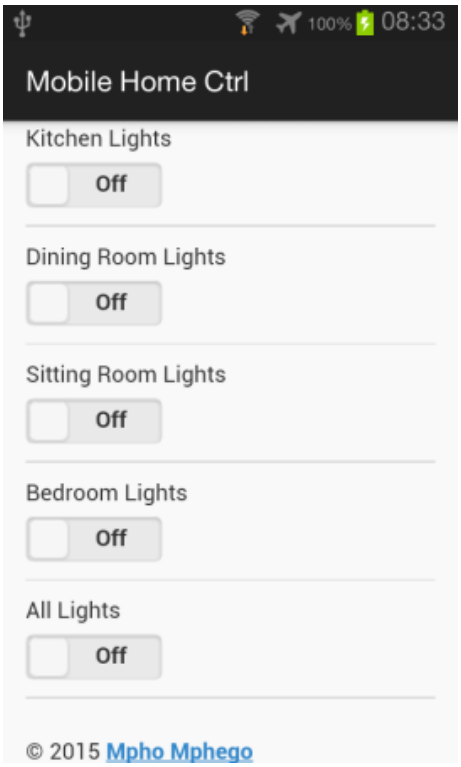


Figure 3-1 Android Control

3.2.2 Closet Door Warning/ Smart Closet

In this section we will discuss as to what we mean by closet door warning or smart closet.
Imagine if your closet or wardrobe could tell you to take a jacket or an umbrella due to the change of outside temperature either current temperature outside or forecasted, well a feature that does exactly that is implemented on the proposed project.

When the user opens the closet or wardrobe, a reed/button switch is triggered. When the system detects a trigger it retrieves the whole days forecast of maximum and minimum temperatures including any chances of rain. If there are low temperatures it notifies the user via the speaker that it will be cold perhaps he/she should take a warm jacket with else if it is going to rain it notifies the user that he/she should perhaps take an umbrella with.

The voice notes a pre-programmed/recorded and can be modified from the configuration file, we use Google's API tts (Text to Speech).
Text to speech, abbreviated as TTS, is a form of speech synthesis that converts text into spoken voice output.
The weather data is retrieved in real-time, considering that the system is connected to the internet, the information is retrieved from www.openweathermap.com using OpenWeatherMaps Python API, OpenWeatherMaps is an online service that provides a free API for weather data, including current weather data, forecasts, and historical data to the developers of web services and mobile applications.

The Figure 3-2 Closet door warning flowchart below, illustrates how this feature of the proposed project works. A script that monitors the closet door switch is executed upon system boot, and it wait for an interrupt on a RPi's GPIO pin, if an interrupt is detected a function that retrieves the weather is executed, the weather information gets retrieved in real-time from OpenWeatherMaps, if rain or cold is read from the retrieved data in this case a JSON file, a notification by means of voice is read out loud from the systems speakers notifying the user if perhaps he/she should carry and umbrella or jacket.

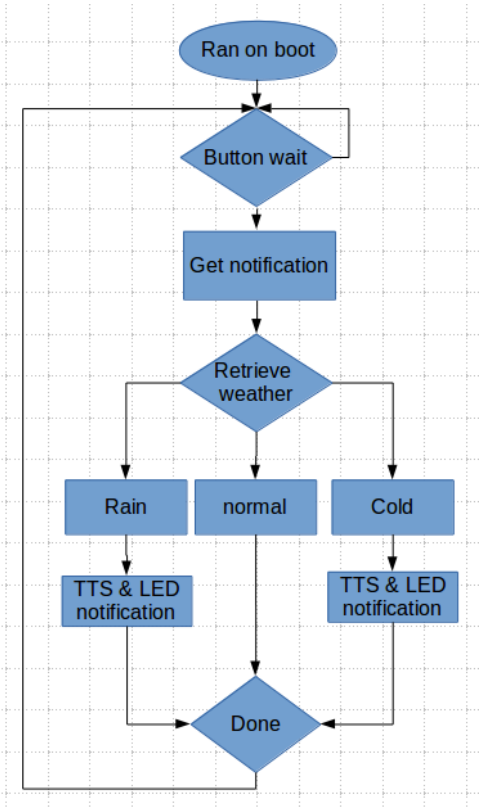


Figure 3-2 Closet door warning flowchart

3.2.3 Gesture Control

The gesture control consist of two (2) features, namely Voice Recognition and Mobile Shake control. Figure 3-3 Mobile Gesture Control dialog below shows the dialog box presented upon application launch.

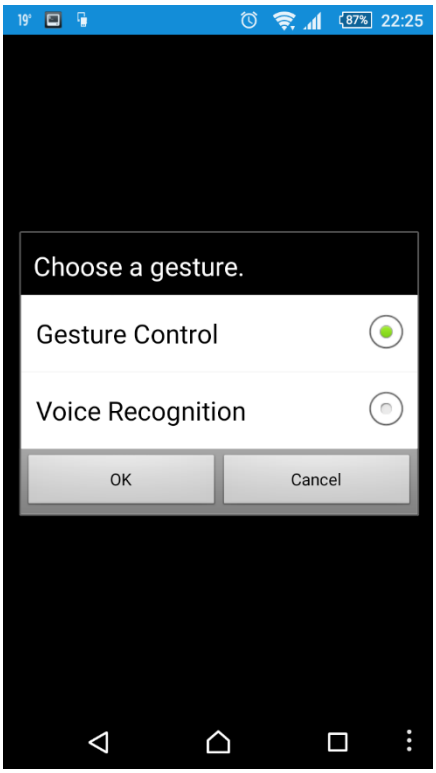


Figure 3-3 Mobile Gesture Control dialog

3.2.3.1 Voice Recognition

Voice/Speech recognition is the ability of a machine or program to identify words and phrases in spoken language and convert them to a machine-readable format (binary 1 and/or 0). Basic speech recognition software has a limited vocabulary of words and phrases and may only identify these if they are spoken very clearly.

3.2.3.1.1 Front End

By exploiting the Google's Android API's we can have access to the low-level controls of an Android ran smartphone and have control of low-level Offline Google Android application. In this paragraph we will explain how the Android voice recognitions works.
The Android speech recognizer uses Natural language processing.

What is NLP?
Natural language processing (NLP) is the ability of a computer program to understand human speech as it is spoken. NLP is a component of artificial intelligence (AI).

There are a couple of layers in processing speech. First Google tries to understand the consonants and the vowels. That is the foundational layer. Next, it uses those to make intelligent guesses about the words. And then higher.

The same approach is actually applied to image analysis where you try to first detect edges in an image. Then check for edges close to each other to find a corner. Then go higher from there.

Figure 3-4 Block diagram speech recognition
below illustrates how Speech recognition works, however we will not go into details as this is low level artificial intelligences using natural language processing and pattern recognition/matching.

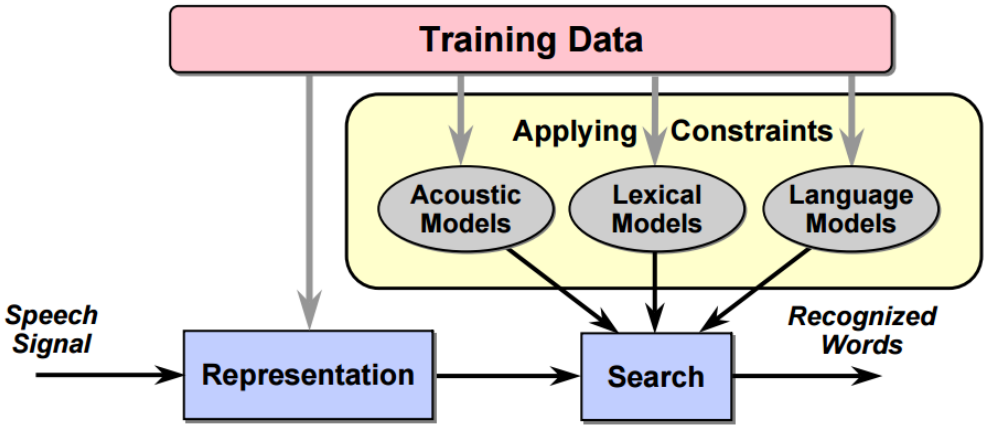


Figure 3-4 Block diagram speech recognition

How does Google's Android recognize speech?

There are four different approaches the recogniser can take when turning spoken words to text:

- Simple pattern matching where each spoken word is recognized in its entirety—the way you instantly recognize a tree or a table without consciously analysing what you're looking at.
- Pattern and feature analysis where by each word is broken into bits and recognized from key features, such as the vowels or constant it may contains.
- Language modelling and statistical analysis in which a knowledge of grammar and the probability of certain words or sounds following on from one another is used to speed up recognition and improve accuracy – This is evident on Apple Siri where is stores the words on cloud and uses an algorithm to compare and select speech to text.
- Artificial neural networks or natural language processing : Brain-like computer models that can reliably recognize patterns, such as word sounds, after exhaustive training

How does the mobile voice recognition work?
In this section we will breakdown how we exploit Google's API speech to text and how we send the text or string over UDP to the server, in this case a Raspberry Pi listening in on an open port and be able to switch different lights/ plugs in and around the house.

The Figure 3-5 Voice Recognition flowchart below, shows how the Python code is implemented on the mobile device running Android OS.

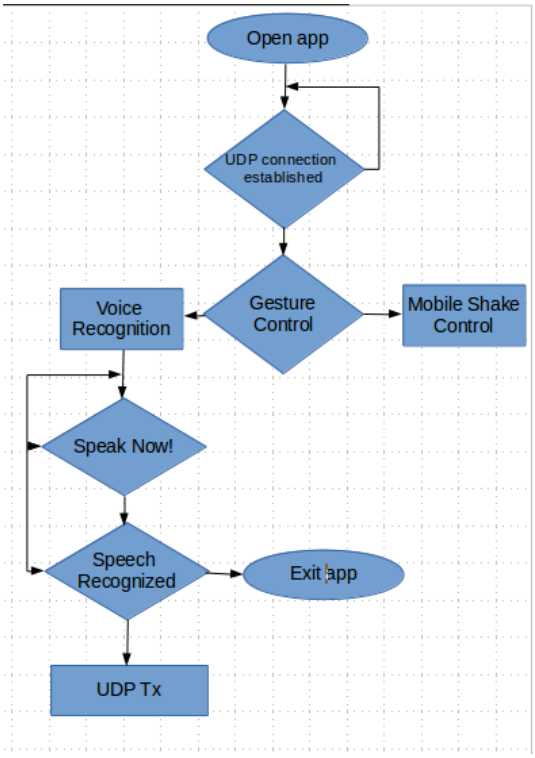


Figure 3-5 Voice Recognition flowchart

When the application is launched a dialog box is displayed offering the user two (2) options - Voice recognition and mobile shake control. When voice recognition option is selected – the application starts broadcasting its UDP connection to all ports open on the entire subnet network, if the Raspberry Pi accepts the UDP connection then the Android application requests the user to input voice commands (Speak Now), Figure 3-6 Voice Recognition Speak request below shows the pop up,

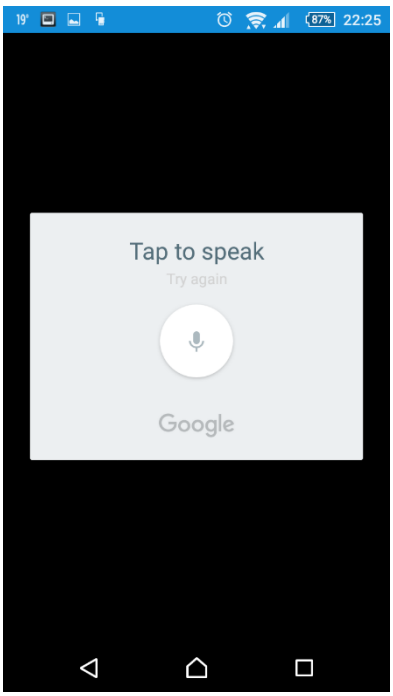


Figure 3-6 Voice Recognition Speak request

If words are recognized their sent over to the
server for more processing, else if the words are not clearly recognized it will request user to input voice commands once more this is done in a loop until the command ‘Exit’ is received.

3.2.3.1.2 Back End

The Raspberry Pi runs a script that constantly listen for UDP packets on a certain port, this script is executed upon system reboot with an exception of restarting the program in-case of any runtime failures.
Upon receiving the UDP packets sent from the Android smartphone as a list of strings – voice converted to text via Android’s API, The script evaluates if whether the string is of text or numbers (floats accelerometer data), if the raw data received via UDP are a list of strings their compared to the strings that are already stored on the running script - This strings are programmable via the configuration file.

The program uses pattern recognition – as it listens to specific words or text if the word is invalid this gets logged into a file for analysis on a later stage. The Figure 3-7 Speech-to-Text diagram/schematic below illustrates how this feature of the proposed project is implemented.

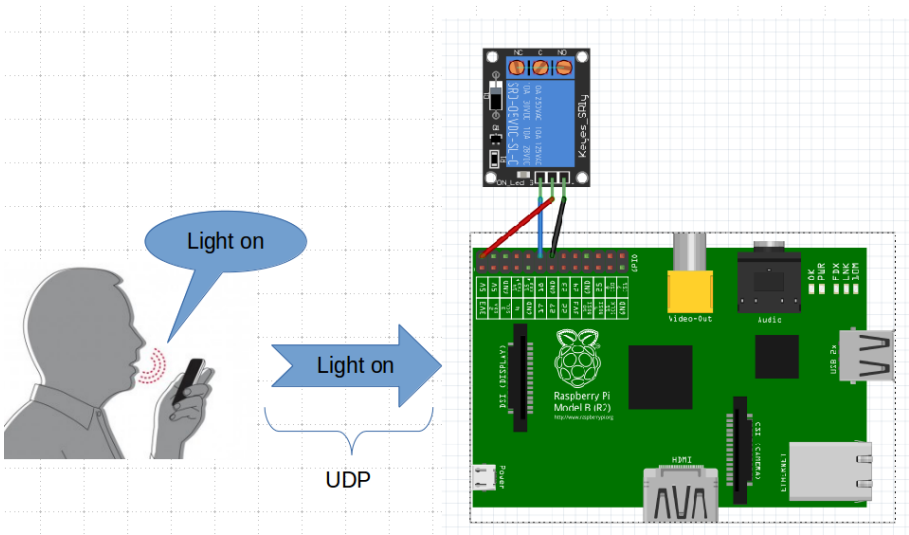


Figure 3-7 Speech-to-Text

When a certain known pattern of word is recognized certain relays are switched either on or off. The Python program below is executed when a string of words are recognized and depending on the word matched, this will execute a relay control command.

```
def voice_recognition(data):
    if data == "bedroom light on" or data == "bedroom on":
        relay_on(Relay1)
        print data
        LOGGER.info('Data: {}'.format(data))

    elif data == "bedroom light off" or data == "bedroom off":
        relay_off(Relay1)
        print data
        LOGGER.info('Data: {}'.format(data))

    elif data == "kitchen light on" or data == "kitchen on":
        relay_on(Relay2)
        print data
        LOGGER.info('Data: {}'.format(data))

    elif data == "kitchen light off" or data == "kitchen off":
        relay_off(Relay2)
        print data
        LOGGER.info('Data: {}'.format(data))

    elif data == "dining light on" or data == "dining light off":
        relay_on(Relay3)
        print data
        LOGGER.info('Data: {}'.format(data))

    elif data == "dining light off" or data == "dining off":
        relay_on(Relay3)
        print data
        LOGGER.info('Data: {}'.format(data))

    elif data == "tv room light on" or data == "tv room light off":
        relay_on(Relay4)
        print data
        LOGGER.info('Data: {}'.format(data))

    elif data == "tv room light off" or data == "tv room off":
        relay_off(Relay4)
        print data
        LOGGER.info('Data: {}'.format(data))
   elif data == "all lights on" or data == "lights on":
        for relay, gpio in relays_conf.iteritems():
            relay_on(gpio)
        print data
        LOGGER.info('Data: {}'.format(data))
    elif data == "all lights off" or data == "lights off":
        for relay, gpio in relays_conf.iteritems():
            relay_off(gpio)
        print data
        LOGGER.info('Data: {}'.format(data))
    else:
        print "invalid parameter: ", data

while True:
# listen to UDP
    data, addr = sock.recvfrom(nbytes)
    """ format
        data : str containing list
        addr : 'xxx.xxx.xxx.xxx'
        print 'data length', len(data)
    """
    try:
        eval(data)
        gesture_control()
    except Exception:
        voice_recognition(data)
```

3.2.3.2 Mobile shake control

3.2.3.2.1 Front End

By exploiting the Google's Android API's we can have access to the low-level controls of an Android ran smartphone and retrieve sensor data. In this paragraph we will explain how the mobile shake control works. The Android mobile platforms supports various sensor. Such as:

- Motion sensors: measure acceleration forces and rotational forces along three axes (in most devices) [e.g.: accelerometers, gravity sensors, gyroscopes and rotational vector sensors]
- Position sensors: measure the physical position of a device [e.g.: orientation sensors and magnetometers]
- Environmental sensors: measure environmental parameters (temperature and pressure, illumination, and humidity) (depend on device) [e.g.: Barometers, photometers, and thermometers.]

The proposed project mainly focuses on Motion sensors which is to say accelerometer sensor. We will read raw data using Python Script Layer for Android and transmit them over UDP protocol to the Raspberry Pi for processing.

The Scripting Layer for Android (abridged as SL4A, and previously named Android Scripting Environment or ASE) is a library that allows the creation and running of scripts written in various scripting languages directly on Android devices. SL4A is designed for developers and (as of March 2016) is still alpha quality software. [4]

The Figure 3-8 Flow chart mobile shake control below, shows how the Python code is implemented on the mobile device running Android OS. When the application is launched a dialog box is displayed offering the user two (2) options - Voice recognition and mobile shake control, upon selecting mobile shake control the logic on the flow chart is used.

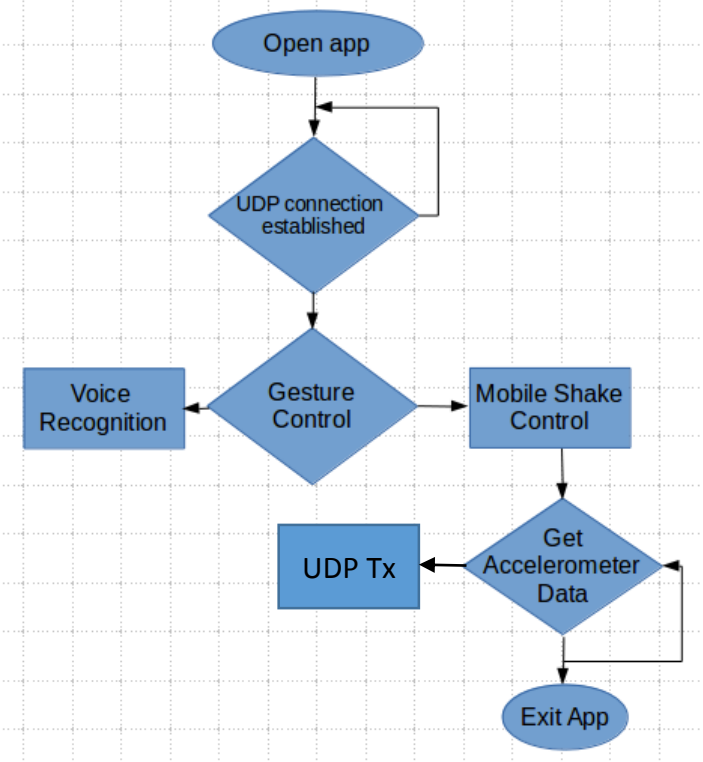


Figure 3-8 Flow chart mobile shake control

When the mobile shake control option is selected the application starts broadcasting its UDP connection to all ports open on the network, If the Raspberry Pi accepts the UDP connection then the Android application starts sending packets. The packets are a string of raw accelerometer (X, Y and Z coordinates see Figure 3-9 3-Axis Accelerometer) data read from the Android device at a sampling rate of 100ms. This activity is done every 500ms until the user decides to exit the application.
There are no calculations done at this point.

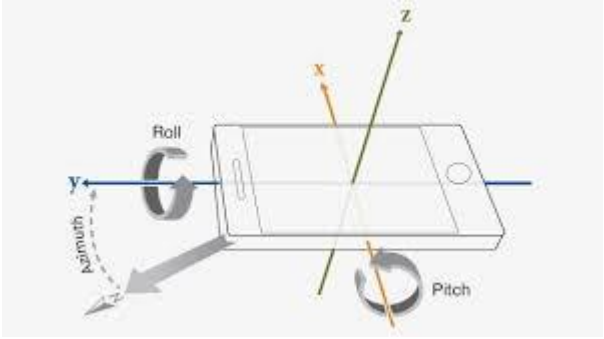


Figure 3-9 3-Axis Accelerometer

What is an accelerometer?
An accelerometer is a device that measures proper acceleration ("g-force"). Proper acceleration is not the same as coordinate acceleration (rate of change of velocity). For example, an accelerometer at rest on the surface of the Earth will measure an acceleration g= 9.81 m/s^2 straight upwards. By contrast, accelerometers in free fall (falling toward the center of the Earth at a rate of about 9.81 $/s^2$) will measure zero. [5]

How to measure acceleration?

The accelerometer in the mobile device provides the XYZ coordinate raw values, which is used to measure the position and the acceleration of the device. The XYZ coordinate represents direction and position of the device at which acceleration occurred. The rotation direction and position are measured using gyroscope sensors that are found in the Android devices. The mobile device rest in the Earth includes the acceleration due to gravity (g = 9.81m/s^2) and the acceleration value. The accelerometer values provided by the device normally includes the gravity as well. Accelerometer along with the linear acceleration and gyroscope will provide results close to accuracy. Linear acceleration does not include the gravity.

3.2.3.2.2 Back End

The Raspberry Pi runs a script that constantly listen for UDP packets on a certain port, this script is executed upon system reboot with an exception of restarting the program in-case of any runtime failures.
Upon receiving the UDP packets sent from the Android smartphone as a string with 3 raw accelerator sensor, the raw sensor data from the x, y, and z axes are analysed and used to check the presence of movements. Here, we stabilize the sensor data values by filtering each value from the x, y, and z axes through the Low-pass filter. The resulting values are compared with the pre-selected Threshold value to judge the presence of movements on each axis. The highest value among the judged values is the directional value for the gesture. Then, we see if there is shake by integrating data values on each axis. We need to set up interval timings for measurements and also need to determine the threshold value to judge movements as well as shake information

```
def gesture_control():
    #print 'Accelerometer: ',data
    x_data, y_data, z_data = eval(data)
    # Sleep for every samples.
    time.sleep(sleep_time)
    Current_Acc = 0.0
    Prev_Acc = 0.0
    if x_data is not None:
        # https://en.wikipedia.org/wiki/Low-pass_filter
        # simple Low pass filter algorithm
        Last_Acc = Current_Acc
        Current_Acc = np.abs((float(x_data ** 2 + y_data ** 2 + z_data ** 2)))
        delta = Current_Acc - Last_Acc
        Prev_Acc = Prev_Acc + alpha * delta
        if Prev_Acc >= sensitivity <= limit:
            LOGGER.info ('Mobile Shaken: Relay ON')
            print ('Mobile Shaken: Relay ON')
            gesture_switch_on()
            if rst_counter > 1:
                LOGGER.info ('Mobile Shaken: Relay OFF')
                print ('Mobile Shaken: Relay OFF')
                gesture_switch_off()
```

Figure 3-10 Mobile Shake Detection-Python

What is low-pass filter?

A low-pass filter is a filter that passes signals with a frequency lower than a certain cutoff frequency and attenuates signals with frequencies higher than the cutoff frequency. The amount of attenuation for each frequency depends on the filter design.

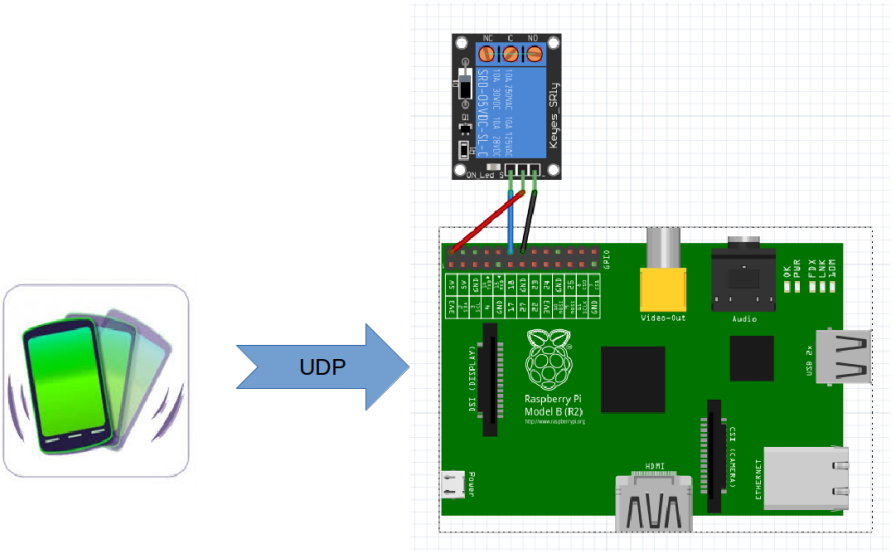


Figure 3-11 Mobile Shake Control

A Python function call, retrieves the 3-axis accelerometer raw data passes it through a low pass filter. Effectively, this “smoothes” out the data by taking out the jittery, high-frequency noise and then compares the magnitude value to the sensitivity and limit value set upon start-up. If the value received detects a mobile switch then a relay is triggered on and if another mobile shake is detected again the relay is triggered off. A basic schematic and android communication is show above Figure 3-11 Mobile Shake Control

Low pass filter algorithm used to derive the results: [6]

```
// Return RC low-pass filter output samples, given input samples,
// time interval dt, and time constant RC
function lowpass(real[0..n] x, real dt, real RC)
  var real[0..n] y
  var real α := dt / (RC + dt)
  y[0] := x[0]
  for i from 1 to n
      y[i] := α * x[i] + (1-α) * y[i-1]
  return y
```

3.2.4 Humidity and Temperature sensor (DHT11)

The Figure 3-12 DHT11 Sensor shows the DHT11 sensor which includes a resistive-type humidity measurement component, an NTC temperature measurement component and a high-performance 8-bit microcontroller inside, and provides calibrated digital signal output. It has high reliability and excellent long-term stability, due to the digital signal acquisition technique and temperature & humidity sensing technology.

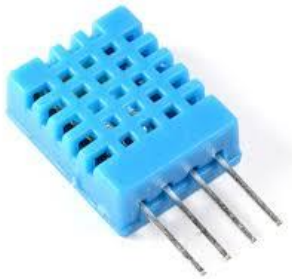


Figure 3-12 DHT11 Sensor

Each DHT11 is strictly calibrated in the laboratory that is extremely accurate on humidity calibration. The calibration coefficients are stored as programmes in the OTP memory, which are used by the sensor's internal signal detecting process. The single-wire serial interface makes system integration quick and easy. Its small size, low power consumption and up-to-20 meter signal transmission.

3.2.4.1 Communication Process: Serial interface

Single bus data format is used for the communication and synchronization between microcontroller and the DHT11 sensor. Each communication process will last about 4ms.

Data consists of decimal and integral parts. A complete data transmission is 40bit, and the sensor sends **higher data bit** first.
Data format:
8bit integral RH data + 8bit decimal RH data + 8bit integral T data + 8bit decimal T data + 8bit check sum.
If the data transmission is correct, the check-sum should be the last 8bit of "8bit integral RH data + 8bit decimal RH data + 8bit integral T data + 8bit decimal T data".

When MCU sends a start signal, DHT11 changes from the low-power-consumption mode to the running-mode, waiting for MCU completing the start signal. Once it is completed, DHT11 sends a response signal of 40-bit data that include the relative humidity and temperature information to MCU. Users can choose to collect (read) some data. Without the start signal from MCU, DHT11 will not give the response signal to MCU. Once data is collected, DHT11 will change to the low power-consumption mode until it receives a start signal from MCU again.

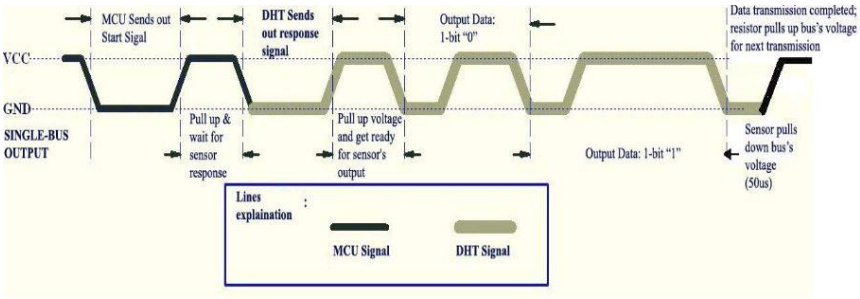


Figure 3-13 DHT11 Communication Process

The default status of the DATA pin is high. When the communication between MCU and DHT11 starts, MCU will pull down the DATA pin for least 18ms. This is called "Start Signal" and it is to ensure DHT11 has detected the signal from MCU. Then MCU will pull up DATA pin for 20-40us to wait for DHT11's response.

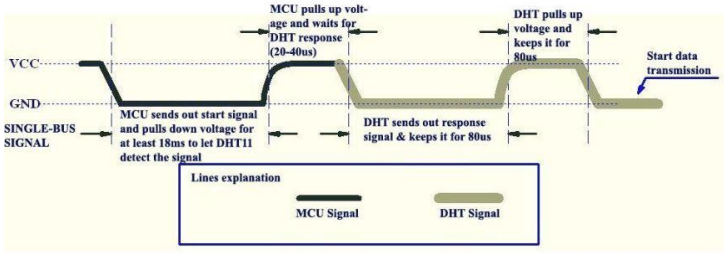


Figure 3-14 MCU sends out Start signal & sensor responses

Once DHT11 detects the start signal, it will pull down the DATA pin as "Response Signal", which will last 80us. Then DHT11 will pull up the DATA pin for 80us, and prepare for data sending.

During the data transition, every bit of data begins with the 50us low-voltage-level and ends with a high-voltage-level signal. The length of the high-voltage-level signal decides whether the bit is "0" or "1".

Data bit "0" has 26-28us high-voltage length:

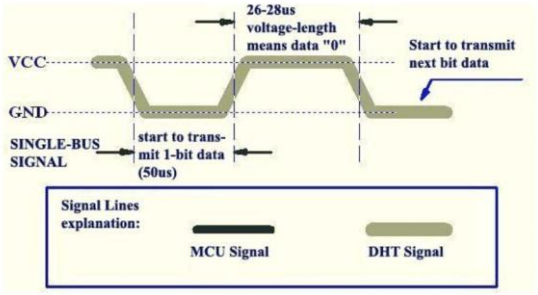


Figure 3-15 Data '0' indication

While data bit "1" has 70us high-voltage length:

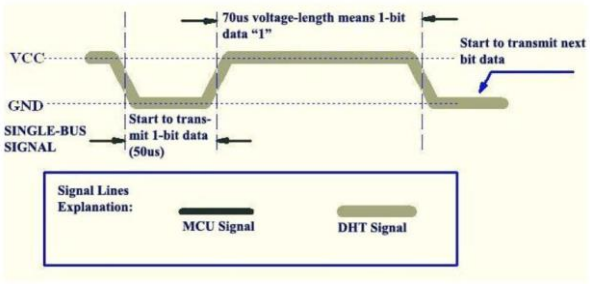


Figure 3-16 Data '1' indication

If the response signal from DHT is always at
high-voltage-level, it suggests that DHT is not responding properly and please check the connection. When the last bit data is transmitted, DHT11 pulls down the voltage level and keeps it for 50us. Then the Single-Bus voltage will be pulled up by the resistor to set it back to the free status.

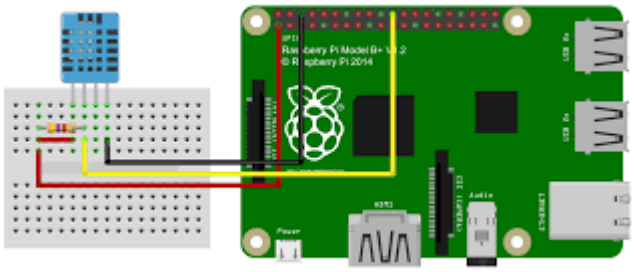


Figure 3-17 DHT11 Circuit

The Figure 3-17 DHT11 Circuit above shows the circuit diagram on how the DHT11 is connected to the Raspberry Pi. To control the DHT11 we use C, as it is a lower level language, it controls the GPIO pin in a more direct way with minimal amount of read failures.

3.2.4.2 C code

```
/*
 *  dht11.c:
 *          Simple test program to test the wiringPi functions
 *          DHT11 test
 */
#include <wiringPi.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#define MAXTIMINGS      85
#define DHTPIN          7
int dht11_dat[5] = { 0, 0, 0, 0, 0 };

void read_dht11_dat()
{
        uint8_t laststate       = HIGH;
        uint8_t counter                 = 0;
        uint8_t j               = 0, i;
        float   f; /* fahrenheit */

        dht11_dat[0] = dht11_dat[1] = dht11_dat[2] = dht11_dat[3] = dht11_dat[4] = 0;

        /* pull pin down for 18 milliseconds */
        pinMode( DHTPIN, OUTPUT );
        digitalWrite( DHTPIN, LOW );
        delay( 18 );
        /* then pull it up for 40 microseconds */
        digitalWrite( DHTPIN, HIGH );
        delayMicroseconds( 40 );
        /* prepare to read the pin */
        pinMode( DHTPIN, INPUT );

        /* detect change and read data */
        for ( i = 0; i < MAXTIMINGS; i++ )
        {
                counter = 0;
                while ( digitalRead( DHTPIN ) == laststate )
                {
                        counter++;
                        delayMicroseconds( 1 );
                        if ( counter == 255 )
                        {
                                break;
                        }
                }
                laststate = digitalRead( DHTPIN );

                if ( counter == 255 )
                        break;

                /* ignore first 3 transitions */
                if ( (i >= 4) && (i % 2 == 0) )
                {
                        /* shove each bit into the storage bytes */
                        dht11_dat[j / 8] <<= 1;
                        if ( counter > 16 )
                                dht11_dat[j / 8] |= 1;
                        j++;
                }
        }

        /*
         * check we read 40 bits (8bit x 5 ) + verify checksum in the last byte
         * print it out if data is good
         */
        if ( (j >= 40) &&
             (dht11_dat[4] == ( (dht11_dat[0] + dht11_dat[1] + dht11_dat[2] + dht11_dat[3]) & 0xFF) ) )
        {
                f = dht11_dat[2] * 9. / 5. + 32;
                printf( "Humidity = %d.%d %% Temperature = %d.%d *C (%.1f *F)\n",
                        dht11_dat[0], dht11_dat[1], dht11_dat[2], dht11_dat[3], f );
        }else  {
                printf( "Data null\n" );
        }
}
```

```
int main( void )
{
	printf( "Raspberry Pi DHT11 Temperature \n" );

	if ( wiringPiSetup() == -1 )
		exit( 1 );

	while ( 1 )
	{
		read_dht11_dat();
		delay( 1000 ); /* wait 1sec to refresh */
	}
	return(0);
}
```

3.2.5 Linux Infrared Remote Control

A part of the proposed system is an infrared controller, controlling heating and cooling inside the house. By using the Linux infrared remote control package it is achievable.

What is IR?
Infrared (IR) light is invisible electromagnetic radiation. Everything absorbs and emits IR, and it's utilised in a plethora of applications.
IR sensors consist of a photocell & chip tuned to look out for specific wavelengths of invisible infrared light. This is why IR is used for remote control detection, such as your TV.

In order to use IR for remote sensing, we need an IR LED (in the remote to output the IR signal) coupled with an IR sensor (inside the TV) which detects the IR pulses and follows the direction that these pulses are coded to e.g. turn off, change channel etc.

What is LIRC (Linux Infrared Controller)?
LIRC is a package that allows you to decode and send infra-red signals of many (but not all) commonly used remote controls.

The most important part of LIRC is the lircd daemon which decodes IR signals received by the device drivers and provides the information on a socket. It also accepts commands for IR signals to be sent if the hardware supports this. The second daemon program called lircmd will connect to lircd and translate the decoded IR signals to mouse movements. You can e.g., configure X11 to use your remote control as an input device.

The user space applications allows you to control your computer with your remote control. You can send X11 events to applications, start programs and much more on just one button press. The possible applications are obvious: Infra-red mouse, remote control for your TV tuner card or CD-ROM, shutdown by remote, program your VCR and/or satellite tuner with your computer, etc. I've heard that using lirc on Raspberry Pie is quite popular these days. [7]

Setting up LIRC on the Raspberry Pi is easy and straight forward, the Figure 3-18 IR Raspberry Pi below shows the basic connection between Raspberry Pi and IR receiver. More details will be described in Chapter 4.

First, we'll need to install and configure LIRC to run on the Raspberry Pi:

sudo apt-get install lirc

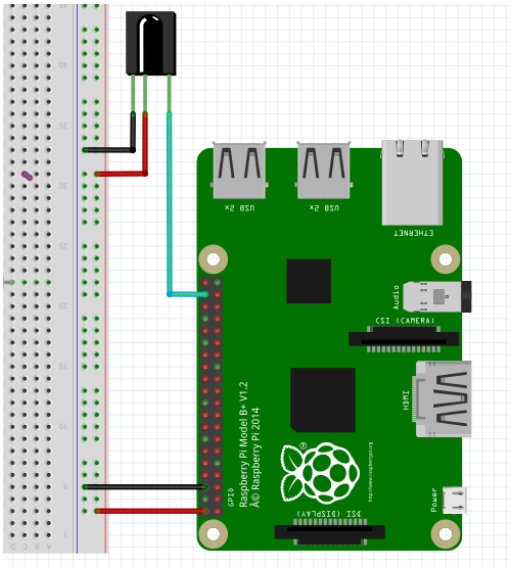


Figure 3-18 IR Raspberry Pi

3.2.6 Light sensor

A photoresistor, or light-dependent resistor (LDR), or photocell is a resistor whose resistance will decrease when incident light intensity increase; in other words, it exhibits photoconductivity.

A photoresistor is made of a high resistance semiconductor. If light falling on the device is of high enough frequency, photons absorbed by the semiconductor give bound electrons enough energy to jump into the conduction band. The resulting free electron conduct electricity, thereby lowering resistance.

Figure 3-19 LDR

Reading the LDR value can be tricky, considering that the Raspberry Pi does not have an ADC (Analogue to Digital Convertor).
All GPIO pins are digital pins, they can only output low and high levels or read high to low levels, this is useful for digital sensors that output digital values, however the LDR is an analogue device. In this case for sensors that outputs a variable resistor value there is a simple solution.
By using an RC circuit, we can be able to use an LDR as a light sensor.

An RC (Resistor-Capacitor) circuit or RC filter is an electric circuit composed of resistors and capacitors driven by a voltage or current source. In this project we will make use of a first order RC circuit, we use it to filter a signal by blocking certain frequencies and letting others pass.
In this case, the resistor and capacitor are connected in series, when a voltage is applied across these components the voltage across the capacitor increases and the time it takes for the voltage to reach 60% of the maximum is equal to the resistance x capacitance. When using an LDR this time is directly proportional to the light level. This time is called the time constant and is given by:

$$t = RC$$

Where t is the time it takes, R is resistance in ohms and C is capacitance in farads. [8]

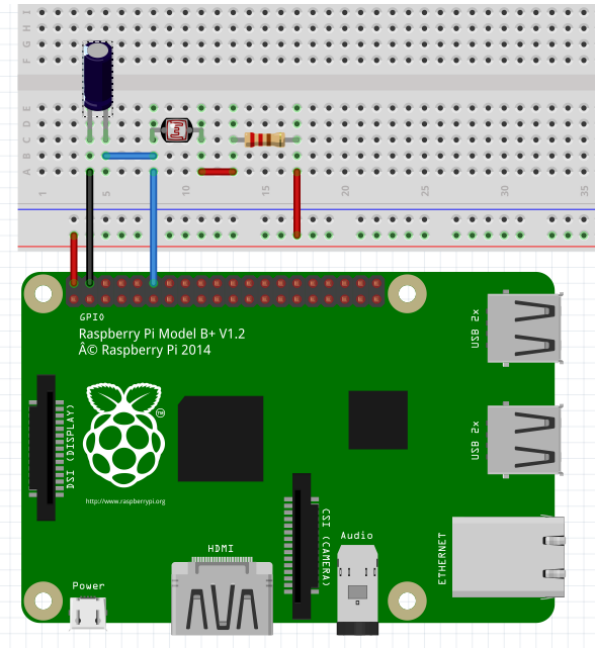


Figure 3-20 LDR sensor

For the LDR to be used as a light level sensor we need to calculate the time it takes for the circuit's voltage to reach a value that will register as a *'HIGH'* on the GPIO input of the Raspberry Pi. Figure 3-20 LDR sensor shows the circuitry.

By means of calculation in order to get a nominal time constant of about 1 second:

$t = RC$
$= (1k\Omega + 1M\Omega) * 1uF$, where 1MΩ is the resistance of LDR in the dark [9]
$= 1$ second

Figure 3-21 LDR reading script, shows the Python code used for reading the light levels and will be explain in details.

```
import RPi.GPIO as GPIO, time, os
from numpy import average

GPIO.setmode(GPIO.BCM)

def RCtime (RCpin):
    reading = 0
    GPIO.setup(RCpin, GPIO.OUT)
    GPIO.output(RCpin, GPIO.LOW)
    time.sleep(0.1)

    GPIO.setup(RCpin, GPIO.IN)
    # This takes about 1 millisecond per loop cycle
    samples = []
    while (GPIO.input(RCpin) == GPIO.LOW):
        reading += 1
        samples.append(reading)
    return average(samples)/10.

def cleanup():
    GPIO.cleanup()
mmphego@SKA-PC:~/Home-Auto-Pi$
```

Figure 3-21 LDR reading script

By setting the GPIO pin as an output and setting it Low, this discharges any charge stored in the capacitor and ensures that both sides of the capacitor have 0v. Setting the GPIO as an input, starts a flow of current through the resistors and through the capacitor to ground and the voltage across the capacitor starts to rise and the time it takes to rise is proportional to the resistance of the LDR.
Monitoring the GPIO pin and reading its value will increment a counter while it waits this value is sampled and averaged out, and at most point the capacitor voltage increases enough to be considered as a *'HIGH'* by the GPIO pin (approximate value calculated above) the time taken is proportional to the light level seen by the LDR, and reset GPIO pin as output and restart the process

3.2.7 Multi Room Media Server

In this section we will discuss the multi room media server feature of the proposed project, but first we need to understand what a media server is.

A media server refers either to a dedicated computer appliance or to a specialized application software, a small personal computer or NAS (Network Attached Storage) for the home, dedicated for storing various digital media (meaning digital videos/movies, audio/music, and picture files).

The proposed system uses UPnP (Universal Plug and Play) protocol.
UPnP (Universal Plug and Play) is a feature that allows the devices on your home network to discover each other and access certain services. Often, this is used for streaming media between devices on a network. Tons of different devices support UPnP streaming nowadays or DLNA (Digital Living Network Alliance), which is great, because it means you can rip or download media to your home computer and stream it directly to your TV - no transfer of files required.

We are using an application called MiniDLNA which can be found in the Ubuntu repository.
MiniDLNA is server software with the aim of being fully compliant with DLNA/UPnP clients. The MiniDLNA daemon serves media files (music, pictures, and video) to clients on a network. Example clients include applications such as totem and xbmc, and devices such as portable media players, smartphones, and televisions.

The proposed project offers the user an option to plug in their external media storage, such as an external USB hard disk drive which contains any form of media from audio, videos to pictures. Upon connecting the disk to the system, a daemon program is instantiated which scans the data and makes it become available on the home network.

This media can be accessed from a Smartphone, Tablet, Smart TV and many other DLNA complaint devices - hence multiroom media server.

Setting up the systems was easy. Below are various commands used to setup the environment.
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install minidlna

After installing the packages, we need to configure it.
sudo nano /etc/minidlna.conf

 # port for HTTP (descriptions, SOAP, media transfer) traffic
port=8200

```
# network interfaces to serve, comma delimited
#network_interface=eth0

# set this to the directory you want scanned.
# * if have multiple directories, you can have multiple media_dir= lines
# * if you want to restrict a media_dir to a specific content type, you
#   can prepend the type, followed by a comma, to the directory:
#   + "A" for audio  (eg. media_dir=A,/home/jmaggard/Music)
#   + "V" for video  (eg. media_dir=V,/home/jmaggard/Videos)
#   + "P" for images (eg. media_dir=P,/home/jmaggard/Pictures)
#media_dir=/opt
media_dir=/mnt
# set this if you want to customize the name that shows up on your clients
friendly_name=Media Server

# set this if you would like to specify the directory where you want MiniDLNA to store its database
and album art cache
db_dir=/var/lib/minidlna

# set this if you would like to specify the directory where you want MiniDLNA to store its log file
log_dir=/tmp/log

# set this to change the verbosity of the information that is logged
# each section can use a different level: off, fatal, error, warn, info, or debug
#log_level=general,artwork,database,inotify,scanner,metadata,http,ssdp,tivo=warn

# this should be a list of file names to check for when searching for album art
# note: names should be delimited with a forward slash ("/")
album_art_names=art.jpg/Art.jpg/front.jpg/Front.jpg/Cover.jpg/cover.jpg/AlbumArtSmall.jpg/al
bumartsmall.jpg/AlbumArt.jpg/albumart.jpg/Album.jpg/album.jpg/Folder.jpg/folder.jpg/Thumb.j
pg/thumb.jpg

# set this to no to disable inotify monitoring to automatically discover new files
# note: the default is yes
inotify=yes

# set this to yes to enable support for streaming .jpg and .mp3 files to a TiVo supporting HMO
enable_tivo=no

# set this to strictly adhere to DLNA standards.
# * This will allow server-side downscaling of very large JPEG images,
#   which may hurt JPEG serving performance on (at least) Sony DLNA products.
strict_dlna=no

# default presentation url is http address on port 80
# presentation_url=http://www.mylan/index.php

# notify interval in seconds. default is 895 seconds.
notify_interval=900
```

```
# serial and model number the daemon will
report to clients
# in its XML description
serial=12345678
model_number=1

# specify the path to the MiniSSDPd socket
#minissdpdsocket=/var/run/minissdpd.sock

# use different container as root of the tree
# possible values:
#   + "." - use standard container (this is the default)
#   + "B" - "Browse Directory"
#   + "M" - "Music"
#   + "V" - "Video"
#   + "P" - "Pictures"
# if you specify "B" and client device is audio-only then "Music/Folders" will be used as root
#root_container=.
```

After the setup, the user is able to plug any external hard drive to the system and stream media contents locally.

3.2.8 Presence Detector & PIR Sensor

3.2.8.1 Passive Wi-Fi detection

Imagine you house could know that you are or not at home or around the vicinity of your house, and certain devices configured by you would switch on or off depending on how the user configured the system.

For example: You are busy in the kitchen using the stove and all of a sudden an emergency at work requires you to immediately rush to the office – while still on the call with your boss you grab your car keys and leave the house – forgetting to switch off the stove behind you.

This is where the presence detector comes in, considering that your mobile devices is always connected to the home network via Wi-Fi. The moment you disconnect from the network the proposed system will disconnect or connect whatever is enabled.

Now think if you had a system of this nature in your home, and involved in a scenario more like the one stated above, you will never have to worry because the system will detect that you have been disconnected from the home network and switch off the stove for you.

By turning the system to a passive Wi-Fi monitoring system we can achieve the above. The detection is done in the system whereby it is constantly searching for a known device, in this case a smart phone.

Figure 3-22 Passive Wi-Fi Detect below, shows the Python script that is executed upon boot. The script assumes that you have initially configured your devices MAC (Media Access Control) address.
A media access control address (MAC address), also called physical address, is a unique identifier assigned to network interfaces for communications on the physical network segment. MAC addresses are used as a network address for most IEEE 802 network technologies, including Ethernet and Wi-Fi. [10]

By using a Linux based network packet scanner (arp-scan), we can be able to determine the devices IP address considering that the system and the mobile device are on the same subnet.
The ARP Scan Tool (also called ARP Sweep or MAC Scanner) is a very fast ARP packet scanner that shows every active IPv4 device on your Subnet. Since ARP is non-routable, this type of scanner only works on the local LAN (local subnet or network segment). [11]

When the IP address is established, the program starts to try to detect the mobile device on the network by using Ping which operates by sending internet control messages protocol (ICMP) *'Echo requests'* to the target in this case a mobile device and waits for an acknowledgement from the mobile device *'Echo Reply'* by measuring the round trip from the transmission we can determine how far the user is from the Wi-Fi hotspot or link.
If a reply is received from a message is logged to the log file that a mobile device is detected and if there user configured the system to perform a specific task upon detection it will be done. After the detection is done the program exits the loop and triggers another instance to rerun the same script again in case the mobile device gets disconnected.

```
# reachable set a  gpio pin and activate flite(voice response) else if it not detected
# gpio will remain off
# Created by Mpho Mphego (mpho112@gmail.com)
# http://pastebin.com/nLkaZ308
import os
import subprocess
import time
from logger import LOGGER
from yamlConfigFile import configFile

try:
    mac_address = configFile()['PresenceDetector']['macaddress']
    if mac_address == str():
        raise KeyError
except KeyError:
    mac_address = 'bc:6e:64:df:d7:d9'

def get_ip(mac_add):
    mob_ip = None
    while True:
        process = subprocess.Popen(
            'sudo arp-scan --interface wlan0 -l | grep {} | cut -f 1'
                .format(mac_add), shell=True, stdout=subprocess.PIPE, )
        mob_ip = process.communicate()[0].strip()
        time.sleep(.5)
        if mob_ip != '':
            break
    return mob_ip

def ping_mob_ip(mobile_ip):
    dev_null = open(os.devnull, 'w')
    if mobile_ip is not None:
        count = 0
        while True:  # Setup a while loop to wait for a button press
            time.sleep(.5)
            if subprocess.call(['ping -c1 {}'.format(mobile_ip)],
                               shell=True,
                               stdout=dev_null,
                               stderr=subprocess.STDOUT) == 0:
                count += 1
                if count > 2:
                    print 'Mobile detected'
                    LOGGER.info("Mobile detected: IP, {}".format(mobile_ip))
                    #subprocess.Popen('mpg123 Welcome_Home.mp3', shell=True, stdout=subprocess.PIPE,)
                    # TODO MM 2015/11/04
                    # Switch on lights
                    count = 0
                    break
        return True

ping_mob_ip(get_ip(mac_address))
mmphego@SKA-PC:~/Home-Auto-Pi/Scripts/PresenceDetector$ a
```

Figure 3-22 Passive Wi-Fi Detect

3.2.8.2 PIR (Passive Infrared) detection

A passive infrared sensor (PIR sensor) is an electronic sensor that measures infrared (IR) light radiating from objects in its field of view. They are most often used in PIR-based motion detectors.

How PIR work?

PIRs are basically made of a pyro electric sensor (which you can see above as the round metal can with a rectangular crystal in the centre), which can detect levels of infrared radiation. Everything emits some low level radiation, and the hotter something is, the more radiation is emitted. The sensor in a motion detector is actually split in two halves. The reason for that is that we are looking to detect motion (change) not average IR levels. The two halves are wired up so that they cancel each other out. If one half sees more or less IR radiation than the other, the output will swing high or low.

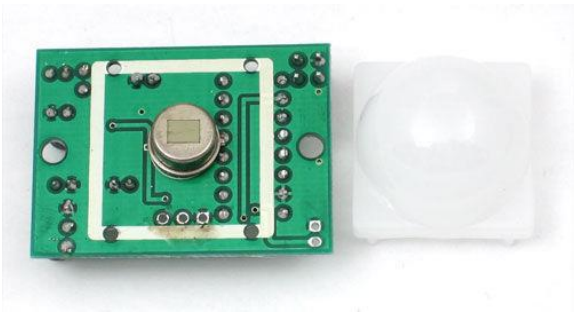


Figure 3-23 PIR

In this part of the proposed systems feature, we use the PIR to detect any motion around the vicinity of the house, this has proved to be a good security feature with the USB camera.

The Figure 3-24 PIR Flowchart below shows the flowchart for the Python script. The script is run automatically on system boot and waits for 10seconds as it is low priority and will run in the background. The motion sensor is configured to run with an interrupt handler, upon motion detection the camera will take an image of the surrounding and sends it to the user via email and push notification as well as log the incident.

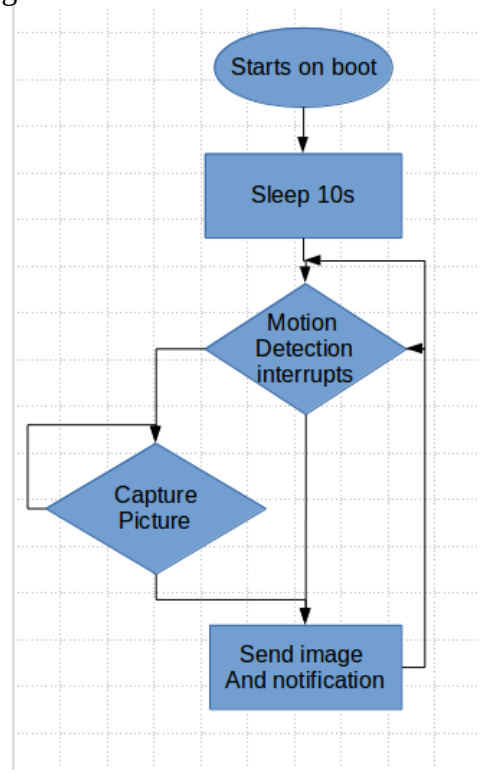


Figure 3-24 PIR Flowchart

3.2.9 Relay Control

In this section we will go into detail as to how the relay switching process works.
This is this the other main feature of the proposed system. This relays are accessed by various functions such as the Gesture control, Closet door warning and many more, depending on the number of relays used – the use if endless.

We are using a 4 way relay module and can be expanded widely by means of using an I/O expanding IC or with an I^2C IC.

The Figure 3-25 Relay Module RPi below shows how it the relay modules are currently connected to the Raspberry Pi and the code Figure 3-26 Relay Control Python below is used to address specific relays to enable or disable them.

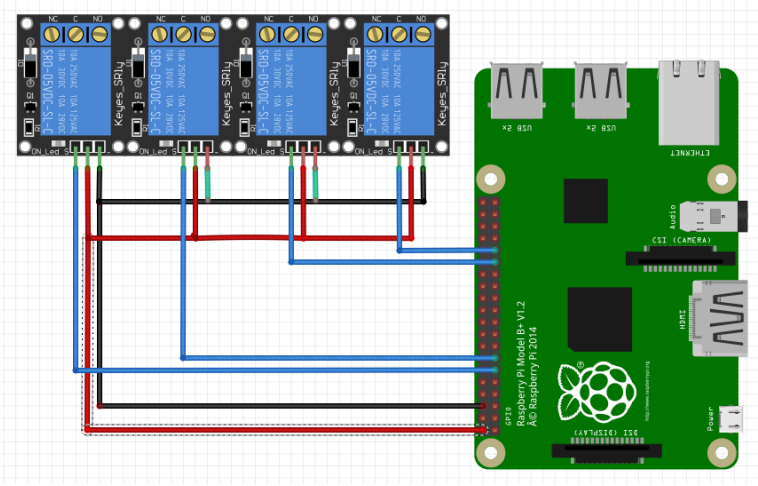


Figure 3-25 Relay Module RPi

```
mmphego@SKA-PC:~/Home-Auto-Pi/Scripts/Relay_Control$ cat relay_control.py
#!/usr/bin/python
import time
import sys
import subprocess

from yamlConfigFile import configFile

relays_conf = configFile()['RelaysSetup']
locals().update(relays_conf)
gpio_control = '/usr/local/bin/gpio'

for relay, gpio in relays_conf.iteritems():
    subprocess.check_call([gpio_control, 'mode', '{}'.format(gpio), 'out'])


def relay_off(pin):
    subprocess.check_call([gpio_control, 'write', '{}'.format(pin), '1'])

def relay_on(pin):
    subprocess.check_call([gpio_control, 'write', '{}'.format(pin), '0'])
```

Figure 3-26 Relay Control Python

3.2.10 Sensor Loggers

In this section of the proposed project, we constantly poll the DHT11 (temperature and humidity sensors), MQ2 (gas sensor connected to an Arduino), LDR (3.2.6 Light sensor), Raspberry Pi's CPU temperature sensors every 30 seconds and uploads the data to **ThingSpeak** (www.thingspeak.com) - **ThingSpeak** is a free web service that lets you collect and store sensor data in the cloud and develop IoT applications.
If certain values retrieved from a sensor are retrieved a push notification is sent to the end user, for example an increase in temperature or increase in gas levels.

3.2.10.1 Python code:

The Python code below is used to sensor poll the sensors and uploads the data, it is ran on system boot, and the script integrates multiple python scripts most where explained on chapter 3.

I will briefly explain the Python sensor polling code and what it does.

To begin we import the Adafruit DHT package that reads the DHT11 sensor, we also import the time package so we're able to put the script to sleep for when we need to and also import the urllib2 package provides an updated API for using internet resources identified by URLs. The gc package provides an interface to the optional garbage collector, the os package provides a portable way of using operating system dependent functionality, the psutil package is a cross-platform library for retrieving information on running processes and system utilization (CPU, memory, disks, network) in Python and sys package provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter.

```
try:
    import Adafruit_DHT as dht
except ImportError:
    import pip
    pip.main(['install', 'Adafruit_DHT'])
import time
import urllib2
import gc
import os
import psutil
import sys
```

We then import Logger package which is responsible for logging useful information to a CSV file, yaml package provides features for reading configuration files and Notifier package which is used to enable push notifications provided an API key.

```
from logger import LOGGER
from yamlConfigFile import configFile
from Notifier import Notification

sys.path.insert(1, '/home/pi/Scripts/Smoke_Detection/')
sys.path.insert(1, '/home/pi/Scripts/Light_sensor/')
```

We then import Smoke detection package which is responsible for reading serial output from the Arduino if it is connected to the Raspberry Pi via USB, ldr_sensor package which is responsible for polling light sensor and retrieve luminosity levels and also GPIO clean-up for garbage collection

```
import Smoke_Detection
from ldr_sensor import RCtime
from ldr_sensor import cleanup
```

```
#cleanup()
wait_time = 30
pin = 26
ldr_pin = 16
```

Thingspeak API URL open source Internet of Things (IoT) application and API to store and retrieve data from things using the HTTP protocol over the Internet or via a Local Area Network.

```
baseurl = 'https://api.thingspeak.com/update?api_key='
```

Variables stored in config file

```
username = configFile()['Email']
password = configFile()['EmailPassword']
send_to = configFile()['Email2']
Notify_api_key = configFile()['PushNotifications']['Pushbullet']
apikey = configFile()['APIs']['thingspeak']
```

A function that retrieves *(3.2.4 Humidity and Temperature sensor (DHT11))* humidity and temperature from DHT11 sensor, in a try/except in-case the function returns null and void.

```
def getSensorData():
    try:
        humid, temp = dht.read_retry(dht.DHT11, pin)
    except:
        return (0, 0)
    return (str(humid), str(temp))
```

A function that reads the smoke detection levels from Arduino via Serial/USB interface, if the USB communication is disconnected the function returns a Zero.

```
def getGasSensorData():
    try:
        if Smoke_Detection.get_gas_sensor() is not None:
            smoke_val = Smoke_Detection.get_gas_sensor()
        else:
            smoke_val = 0
    except:
        smoke_val = 0
    return str(smoke_val)
```

This is where we instantiate the function that reads light levels as explained above *(3.2.6 Light sensor).*

```
def getLightLevels(LDR_pin):
    try:
        if RCtime(LDR_pin) is not None:
            light_val = RCtime(LDR_pin)
        else:
            raise Exception
    except:
        light_val = 0
    return light_val
```

With this function we can retrieve CPU temperature but it is not a Python command but a UNIX command hence we needed to use *'os'* package explained above, if fails to read it returns a Zero.

```
def getCPUTemp():
    try:
        ostemp = os.popen('vcgencmd measure_temp').readline()
        cpu_temp = (ostemp.replace("temp=", "").replace("'C\n", ""))
    except:
        cpu_temp = 0
    return cpu_temp

#---------------------------------------------------------------------

API_URL = baseurl + apikey
count = 0
rst_counter = 0
```

```
sensor_name_1 = 'raspberry_pi_cputemp'
sensor_name_2 = 'DHT11_Temp'
sensor_name_3 = 'DHT11_Humidity'
```

Push notification function used for sending moule notification with the API added above

```
NotifyMe = Notification(username, password, send_to, Notify_api_key)

def notification(alert, message):
    global rst_counter
    rst_counter += 1
    NotifyMe.send_mail(alert, message)
    NotifyMe.send_pushbullet(alert, message)

def notification_check():
    global rst_counter
    rst_counter = 0

try:
```

Here we constantly poll the sensor functions every 30 sends.

```
    while True:
        smoke = getGasSensorData()
        light = getLightLevels(ldr_pin)
        humidity, temperature = getSensorData()
        cpu_temp = getCPUTemp()
```

Storing all retrieved sensor values to CSV file

```
        LOGGER.info('Humidity: {}%, Temp: {}, CPU_Temp: {}, Smoke_Level: {}, Light_Levels: {}'.format(
            humidity, temperature, cpu_temp, smoke, light))

        print ('Humidity: {}%, Temp: {}, CPU_Temp: {}, Smoke_Level: {}, Light_Levels: {}'.format(
            humidity, temperature, cpu_temp, smoke, light))
```

Testing if whether temperature is over 35 degrees, and if it is we send a push notification with a message, same as with the humidity and smoke levels.

```
        #if float(temperature) > 35.:
            #alert = 'Temperature Nofication'
            #message = 'Temperature is at {}degrees'.format(temperature)
            #notification(alert, message)
            #if rst_counter > 1:
                #notification_check()

        #if float(humidity) > 50.:
            #alert = 'Humidity Nofication'
            #message = 'Humidity is at {}%'.format(humidity)
            #notification(alert, message)
            #if rst_counter > 1:
                #notification_check()

        #if float(smoke) > 450.:
            #alert = 'Smoke Nofication'
            #message = 'Smoke is at {}'.format(smoke)
            #notification(alert, message)
            #if rst_counter > 1:
                #notification_check()

        try:
```

In this section we uploading data retrieved from the sensors to Thingspeak API with the URL provided above, if the is no internet connectivity it raises an exception which closes the connection and starts a counter which waits for 5*30seconds and disconnects.

Figure 3-27 Light levels ThingSpeak shows the
results from Thingspeak and one can easily notice that the luminosity levels increased dramatically between the hours of 6pm to 8pm and someone switch the lights on from 8pm till around 12pm then lights were off.

```
            send_data = urllib2.urlopen(API_URL +
                '&field1={}&field2={}&field3={}&field4={}&field5={}&field6={}&field7={}'.format(
                humidity, temperature, cpu_temp, 0,0, smoke, light))
        except:
            count += 1
            send_data.close()
            if count > 5:
                count = 0
```

Waiting time between polls to decrease CPU usage.

```
        time.sleep(wait_time)
        send_data.close()
```

Forcefully collection all garbage's as in unused variables and etc.

```
        gc.collect()
except:
```

An exception that cleans up the GPIO pins in-case of any failures when reading or uploading the data.

```
    cleanup()
```

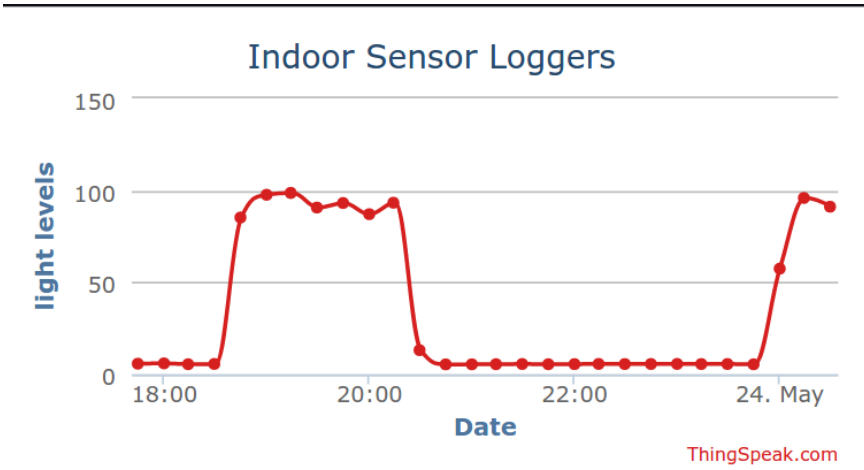


Figure 3-27 Light levels ThingSpeak

https://thingspeak.com/channels/103450/charts/7?average=15&bgcolor=%23ffffff&color=%23d62020&dynamic=false&results=700&type=spline

3.2.11 Smart Alarm

Imagine instead of hearing a depressing continuous beeping meant to wake you up in the morning, Instead of beeping it wakes up with ambient noise and then starts reads out the current time which is followed by the current weather outside and the forecasting for the day. After then forecast it retrieves current headline news from www.news24.com and reads them out loud, all this is done while the systems enables the coffee maker in order for the user to get a steaming cup of coffee before he/she departs from work.

The code is written in Python, and we make use of rss feed parser to access rss feeds from www.news24.com and www.openweathermaps.com to access the weather, Google tts API for the audio. The code is straight forward in summary it retrieves all this data from the above mentioned websites construct a string and parse it to Google while iterating through the words, Google's tts API then translates the text to speech, it then downloads the audio as mp3 files while saving them to our ram disk to preserve the life expectancy of the SD card. After all voice notes are retrieved from Google they are played over the speakers while a relay is activated to switch on the coffee maker and after certain period of time the relay is switched off.

```
__author__ = "Mpho Mphego"
__version__ = "$Revision: 1.3 $"
__description__ = "Voice enabled Smart Alarm with weather, news and coffee notifier"
__date__ = "$Date: 2015/01/31 14:55 $"
__copyright__ = "Copyright (c) 2015 Mpho Mphego"
__url__ = "mpho112.wordpress.com"
__license__ = "Python"

import subprocess
import time
import feedparser
#import RPi.GPIO as GPIO
from better_spoken_time3 import gmt, day
from get_url_weather9 import wtr, frc
from get_url_news8 import news

try:
    import gtts
except ImportError:
    import pip
    pip.main(['install', 'gTTS'])
finally:
    from gtts import gTTS
#coffeemaker = 4 #GPIO0
end = " That's all for now. Have a nice day."
# url = 'http://feeds.feedburner.com/brainyquote/QUOTEBR'
url_quote = 'https://www.quotesdaddy.com/feed/tagged/Inspirational'
rss = feedparser.parse(url_quote)
#GPIO.setwarnings(True)
#GPIO.setmode(GPIO.BCM)
#time.sleep(0.01)
#GPIO.setup(coffeemaker, GPIO.OUT)
#time.sleep(0.01)
# Turn all of the parts into a single string

if rss['status'] == 200:
    quote = '. And todays quote: ' + str(rss['entries'][0]['summary'])
else:
    quote = ''

try:
```

```
    words = str(gmt + day + wtr + frc + news + quote + end)
except UnicodeEncodeError:
    words = str(gmt + day + wtr + frc + news.encode('ascii', 'ignore') + quote + end)
# strip any quotation marks
words = words.replace("'", '').strip().split('. ')

try:
    for i,line in enumerate(words):
        tts = gTTS(text=line, lang='en')
        tts.save('{}.mp3'.format(i))
    # Play the mp3s returned
    [subprocess.call ('mpg123 {}.mp3 '.format(mp3_file), shell=True)
                    for mp3_file in range(i)]

# festival is now called in case of error reaching Google
except subprocess.CalledProcessError:
    subprocess.check_output("echo " + words + " | festival --tts ", shell=True)

# Cleanup any mp3 files created in this directory.
subprocess.call ('sudo rm *.mp3', shell=True)

# Enabling GPIO for relay switch to turn on coffee maker
#GPIO.output(coffeemaker, True)
# Time can be dependent on the make and model of the coffee maker.
#time.sleep(600)
#GPIO.output(coffeemaker, False)
```

3.2.12 Smart Doorbell

A classic doorbell can be defined as a signalling device typically placed near an entry door to a building. When a visitor presses a button the bell rings inside the building, alerting the occupant to the presence of the visitor.

The proposed smart doorbell feature of the project, works in that manner however when a visitor presses the doorbell button it notifies the occupant with an images of the visitor and also sends an email, sms and push notification (to an smart device) to the occupant in case they are not able to hear the bell ringing.

The Figure 3-28 Smart Doorbell flowchart illustrate how the Python code architecture. On boot up the script is activated, an interrupt on a GPIO is a set which waits for the individual to press the button, by making use of interrupts instead of polling a GPIO pin decreases the amount of resources used.
If the button is pressed a pre-recorded voice notification is played for the occupant to check the door as there might be someone, this happens concurrently with an sms notification while a picture of the visitor is captured and sent to both email and push notification. Initially there was a feature to enable 2-way real-time video communication but due to network latency and high resource usage the feature was deprecated.

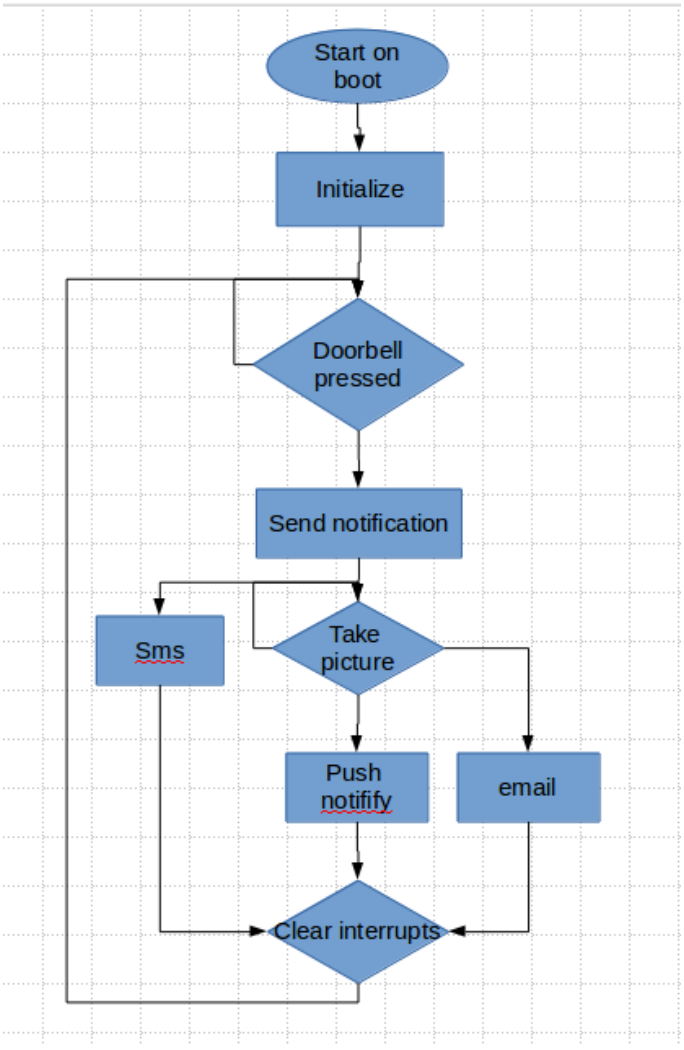


Figure 3-28 Smart Doorbell flowchart

3.2.13 Smoke Detection

Due to the limitation of the Raspberry Pi when it comes to measuring analogue signals, we have resorted in using an additional microcontroller board, the Arduino Uno R3.
The reasoning looks at expanding the capabilities of the proposed systems, but having additional GPIO pins, ADC and I2C to use. The current setup is configured primarily for retrieving gas levels and sending them to the Raspberry Pi – More work on the Arduino will be implemented on a later stage.

What is a Smoke/Gas sensor?
A gas sensor is a transducer that detects gas molecules and which produces an electrical signal with a magnitude proportional to the concentration of the gas.
Unlike other types of measurement, types that are relatively straightforward and deal with voltage, temperature, and humidity, the measurement of gases is much more complicated. Because there are literally hundreds of different gases, and there is a wide array of diverse applications in which these gases are present, each application must implement a unique set of requirements.

The proposed project makes use of a MQ2 gas sensor which is connected to an Arduino.

The MQ series of gas sensors use a small heater inside with an electro-chemical sensor. They are sensitive for a range of gasses and are used indoors at room temperature. Figure 3-29 MQ2 Gas Sensor shows a MQ-2 electro-chemical gas sensor used in this project.

They can be calibrated more or less sensitive but a known concentration of the measured gas or gasses is needed for that.

The output is an analogue signal and can be read with an analogue input of an Arduino or ADC. [12]

Figure 3-29 MQ2 Gas Sensor

Figure 3-30 MQ2 sensitivity graph shows the typical sensitivity characteristics of the MQ-2 for several gases.
In their: Temp: 20 degree C
Humidity: 65%
O2 concentration: 21%
RL=5k
Ro: sensor resistance at 1000ppm of H2 in the clean air.
Rs: sensor resistance at various concentrations of gases.

According to the graph, we can see that the minimum concentration we can test is 100ppm and the maximum is 10000ppm, in other word, we can get a concentration of gas between 0.01% and 1%. However, we can't provide a formula because the relation between ratio and concentration is nonlinear.

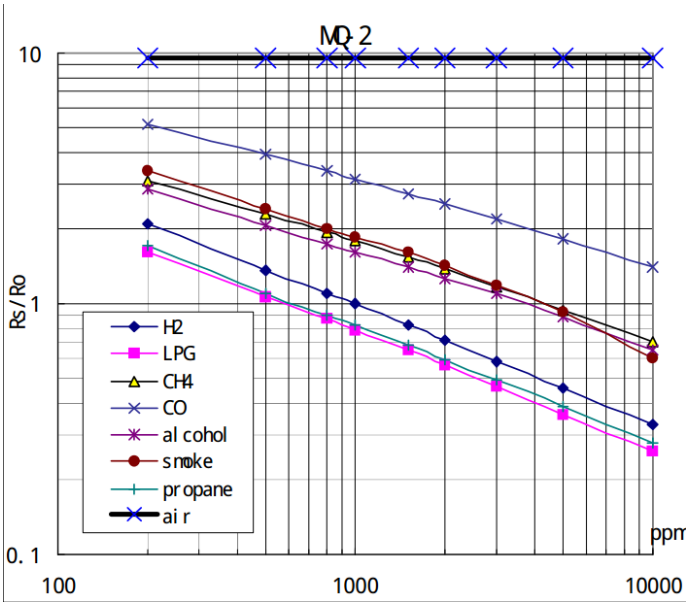


Figure 3-30 MQ2 sensitivity graph

Figure 3-31 Arduino MQ2 below shows how the gas sensor is configured on the Arduino, the Arduino is then connected to the Raspberry Pi via USB interface

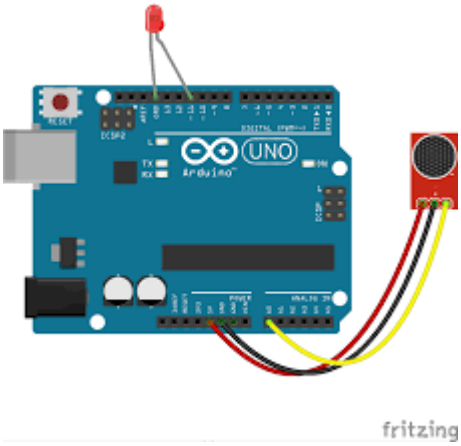


Figure 3-31 Arduino MQ2

3.2.14 TV Proximity Sensor

The aim of this feature of the proposed project is to provide a way that prevents children from watching TV from a close distance to safeguard their eyes health.
The system uses an ultrasonic sensor to determine distance of not more than 40cm away from the TV, If a child is standing in front of the TV for a limited amount of time they will get a warning voice note, that they should move away from the TV, if they do not move away in the amount of time configured the TV will switch off until they move away.

What is an ultrasonic sensor?
Active ultrasonic sensors generate high-frequency sound waves and evaluate the echo which is received back by the sensor, measuring the time interval between sending the signal and receiving the echo to determine the distance to an object. [13]

The Figure 3-32 HC-SR04 shows an ultrasonic sensor which uses sonar to determine distance to an object like bats or dolphins do. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package. From 2cm to 400 cm. It operation is not affected by sunlight or black material like Sharp rangefinders are (although acoustically soft materials like cloth can be difficult to detect). It comes complete with ultrasonic transmitter and receiver module.

Figure 3-32 HC-SR04

The Figure 3-33 TV Proximity Flowchart shows the Python and C script meant to control the proximity sensor. The reason why the sensor sampling is written in C language is because Python is slow as compared to C in terms of execution and running.
On system boot the script is executed which then reads configured variables such as sampling size and rate, wait time and etc. After the initialisation a function that reads the distance and compares it with a pre-set threshold goes on an endless loop testing whether the is someone in front of the TV, if there is, it sends out a notification and waits for a pre-set time.
If the child is still in front of the TV after the notification is executed, the TV is then switched off until the child has stepped away, then it switched back on.

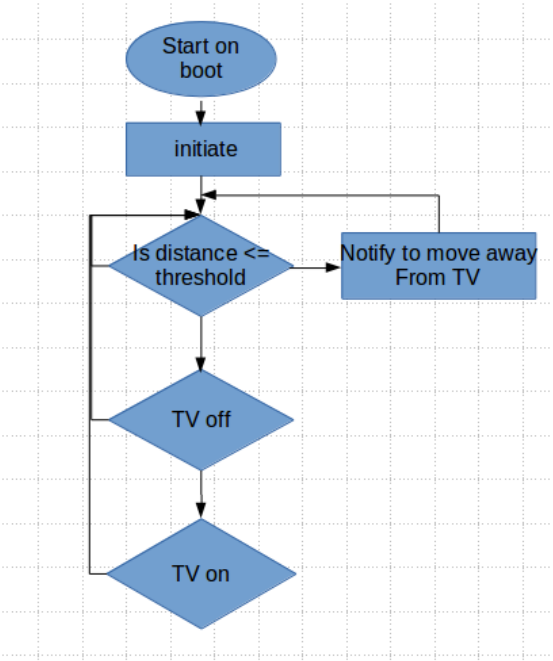


Figure 3-33 TV Proximity Flowchart

The code below was written in C, with the time it takes for the signal to travel to an object and back again, we can calculate the distance using the following formula.

$$Speed = \frac{Distance}{Time}$$

The speed of sound is variable, depending on what medium it's travelling through, in addition to the temperature of that medium. However, some clever physicists have calculated the speed of sound at sea level so we'll take our baseline as the 343m/s.

$$34300 = \frac{Distance}{time/2}$$

$$17150 = Distance/time$$

$$Distance = 17150 * time$$

We also need to divide our time by two because what we've calculated above is actually the time it takes for the ultrasonic pulse to travel the distance to the object and back again. We simply want the distance to the object! We can simplify the calculation to be completed in our C code as follows:

3.2.14.1 C Code

```
#include <stdio.h>
#include <stdlib.h>
#include <wiringPi.h>

#define TRIG 28
#define ECHO 29

void setup() {
	wiringPiSetup();
	pinMode(TRIG, OUTPUT);
	pinMode(ECHO, INPUT);

	//TRIG pin must start LOW
	digitalWrite(TRIG, LOW);
	delay(30);
}
int getCM() {
	//Send trig pulse
	digitalWrite(TRIG, HIGH);
	delayMicroseconds(10);
	digitalWrite(TRIG, LOW);

	//Wait for echo start
	while(digitalRead(ECHO) == LOW);
	//Wait for echo end
	float startTime = micros();

	while(digitalRead(ECHO) == HIGH);
	float travelTime = micros() - startTime;
	//Get distance in cm
	float distance = travelTime * 17150;

	printf ("%f", distance);
	return distance;
}
int main(void) {
	setup();
	delay(0.5);
	getCM();
	return 0;
}
```

3.2.15 Website Interface Control

The Web-control interface is written in both Python and HTML/CSS and it is being hosted on an Apache web server.
Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation. Most commonly used on a Unix-like system (usually Linux), the software is available for a wide variety of operating systems besides Unix, including eComStation, Microsoft Windows, NetWare, OpenVMS, OS/2, and TPF.

The web-server is running on the raspberry pi as the primary web-server.
Figure 3-34 Website Control Interface shows the web-control with graphs listing humidity, CPU temp, indoor temp and gas sensor levels data/sensor logging retrieved from **ThingSpeak API.**

ThingSpeak is an open source Internet of Things (IoT) application and API to store and retrieve data from things using the HTTP protocol over the Internet or via a Local Area Network. ThingSpeak enables the creation of sensor logging applications, location tracking applications, and a social network of things with status updates.

The sensor data is uploaded to the Thingspeak server every 30 seconds intervals.

Main python code for running the webserver. The sensor data logging is done separately and it is independent of the webserver.

The code below is the main webserver.py python script.

```
#!/usr/bin/python2
__author__ = "Mpho Mphego"
__description__ = "Home Automation Webserver and Control Center"
__version__ = "$Revision: 2.0 $"
__date__ = "$Date: 2015/01/22 22:54 $"
__url__ = "mpho112.wordpress.com"
__copyright__ = "Copyright (c) 2015 Mpho Mphego"
__license__ = "Python/HTML"

import sys
import os
from wsgiref.simple_server import make_server
from urlparse import parse_qsl
from html_site import html
from yamlConfigFile import configFile

sys.path.insert(1, '/home/pi/Scripts/Relay_Control/')
from relay_control import *
try:
   locals().update(relays_conf)
except:
   pass

rst_counter = 0
def gesture_switch_on(pin):
      global rst_counter
      rst_counter += 1
      relay_on(pin)
def gesture_switch_off(pin):
```

```
        global rst_counter
        rst_counter = 0
        relay_off(pin)


def application(environ, start_response):
    """Returns a dictionary containing lists as values."""
    d = parse_qsl(environ['QUERY_STRING'])

    try:
        if (d[0][0]=="led1"):
            gesture_switch_on(Relay1)
            if rst_counter > 1:
                gesture_switch_off(Relay1)
        if (d[0][0]=="led2"):
            gesture_switch_on(Relay2)
            if rst_counter > 1:
                gesture_switch_off(Relay2)

        if (d[0][0]=="led3"):
            gesture_switch_on(Relay3)
            if rst_counter > 1:
                gesture_switch_off(Relay3)

        if (d[0][0]=="led4"):
            gesture_switch_on(Relay4)
            if rst_counter > 1:
                gesture_switch_off(Relay4)

        if (d[0][0]=="led5"):
            gesture_switch_on(Relay1)
            relay_on(Relay2)
            relay_on(Relay3)
            relay_on(Relay4)
            if rst_counter > 1:
                gesture_switch_off(Relay1)
                gesture_switch_off(Relay2)
                gesture_switch_off(Relay3)
                gesture_switch_off(Relay4)
                relay_off(Relay1)
                relay_off(Relay2)
                relay_off(Relay3)
                relay_off(Relay4)

    except:
        pass
    response_body = html()
    status = '200 OK'

    # Now content type is text/html
    response_headers = [('Content-Type', 'text/html'),
                ('Content-Length', str(len(response_body)))]
    start_response(status, response_headers)

    return [response_body]

httpd = make_server('0.0.0.0', 8000, application)
# Now it is serve_forever() in instead of handle_request().
# In Linux a Ctrl-C will do it.
#reloader.install()
httpd.serve_forever()
#httpd.handle_request()
```

The above script basically, instantiate a webserver by using a python package called make_server which then creates a webserver on localhost port 8000 which servers the function named 'application'.
The application function consists of an html code, headers and also relay controls by means of using CGI.
Common Gateway Interface (CGI) is a standard way for web servers to interface with executable programs installed on a server that generate web pages dynamically. Such programs are known as CGI scripts or simply CGIs; they are usually written in a scripting language, but can be written in any programming language.

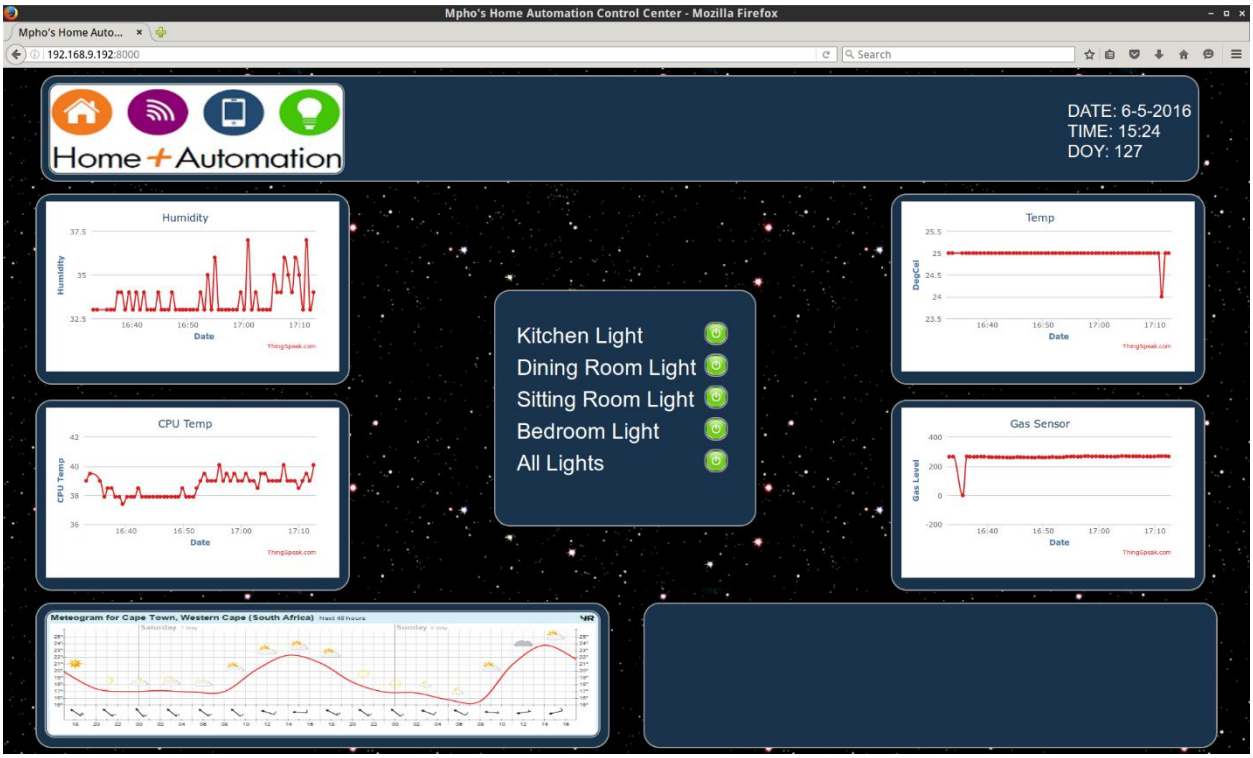


Figure 3-34 Website Control Interface

3.2.16 What's My IP

Any device connected to a Wireless/Local Area Network is assigned an IP address automatically considering that the Raspberry Pi is configured as a **DHCP** client.

What is DHCP?

Dynamic Host Configuration Protocol (DHCP) is a client/server protocol that automatically provides an Internet Protocol (IP) host with its IP address and other related configuration information such as the subnet mask and default gateway.

In order to connect to your Raspberry Pi from another machine using SSH or VNC, you need to know the Raspberry Pi's IP address. This is easy if you have a display connected, and there are a number of methods for finding it remotely from another machine on the network.

What is SSH?

Secure Shell, or SSH, is a cryptographic (encrypted) network protocol operating at layer 7 of the OSI Model (Open Systems Interconnection model) to allow remote login and other network services to operate securely over an unsecured network.

SSH provides a secure channel over an unsecured network in a client-server architecture, connecting an SSH client application with an SSH server. Common applications include remote command-line login and remote command execution, but any network service can be secured with SSH. The protocol specification distinguishes between two major versions, referred to as SSH-1 and SSH-2.
However, considering that the Raspberry Pi will not be connected to a display, finding an IP address becomes difficult. We use the IP address to access the Raspberry Pi remotely either via SSH or webserver to configure, monitor and control.

Hence the development of a Python script that will execute at after every reboot of the Pi. The script retrieves the current IP on the PI and emails it to the users email address in case it has changed from previous more development to the script are still needed for instance a button which a user can just press and then an IP address gets sent to the user's email address.
Figure 3-35 what's my IP below shows an email inbox with the current IP address.

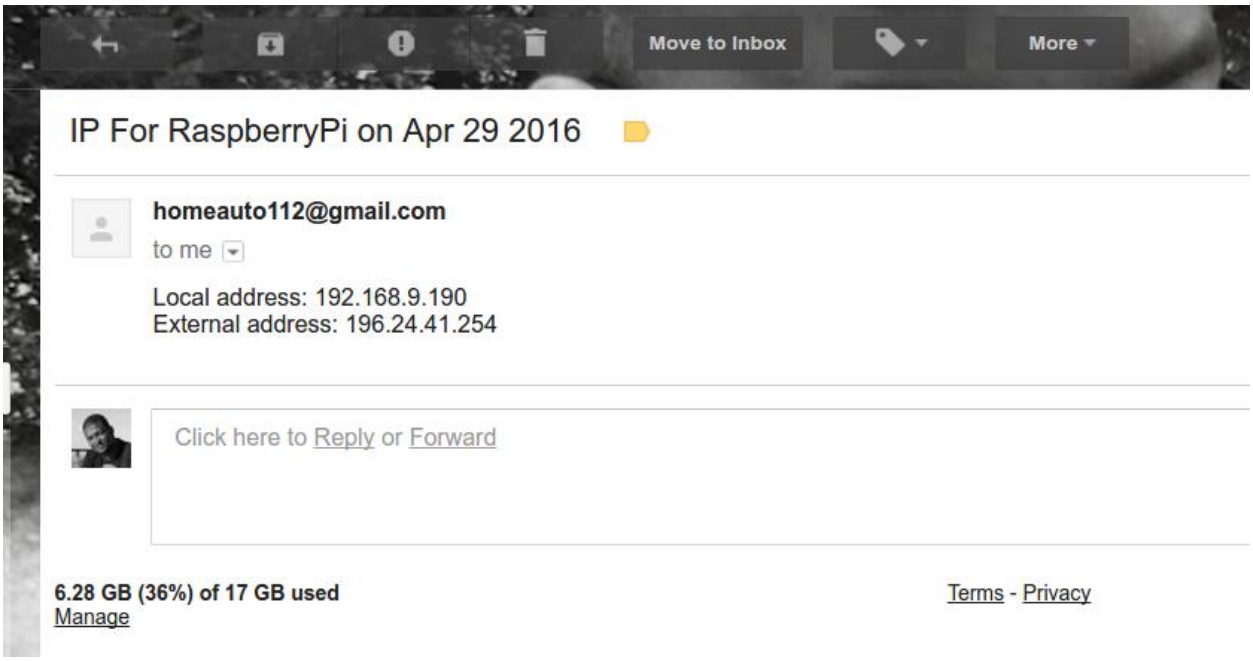


Figure 3-35 what's my IP

By using the Pushbullet's API's we configured the script to also send push notifications to an Android based smartphone.

What is Pushbullet?
Pushbullet is an Android application that allows extensive connectivity between any of your devices, whether that be smartphone to tablet, smartphone to laptop, laptop to tablet, and so on. Through Pushbullet, you can send pictures, links, street addresses, notes and other files between any devices that also have the app.

For example, you take a really awesome photo on your phone that you want to quickly send to your laptop. Or maybe you're reading a web article on your desktop when you suddenly need to leave, and decide to "push" that web address to your smartphone to read on the go. Another great feature of Pushbullet is the ability to mirror notifications from your smartphone to your desktop so you can see alerts from your phone without having it nearby, and then also have the ability to interact with them all from your desktop.

The script can be accessed from: https://github.com/mmphego/Home-Auto-Pi

3.3 Implementation of product / strategy

The Implementation and construction of this project will be implemented in four separate units (stages) connected with the appropriate use of connecting cables and soldered and then mounted into the enclosure. In order to achieve the design, the following separate stages are illustrated.

3.3.1 Relay and Raspberry Pi integration

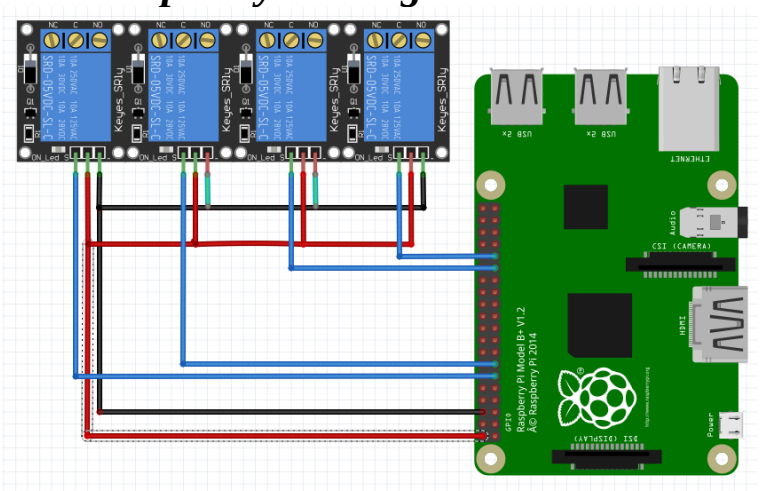


Figure 3-36 Relay and RPi integration

3.3.2 MQ-2 and Arduino integration

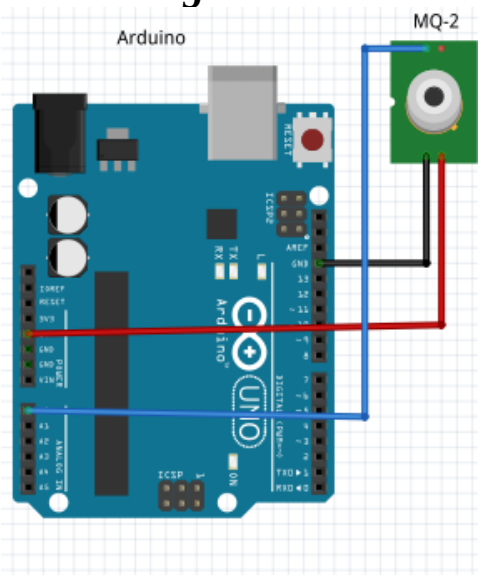


Figure 3-37 MQ-2 and Arduino integration

3.3.3 Sensors and Raspberry Pi integration

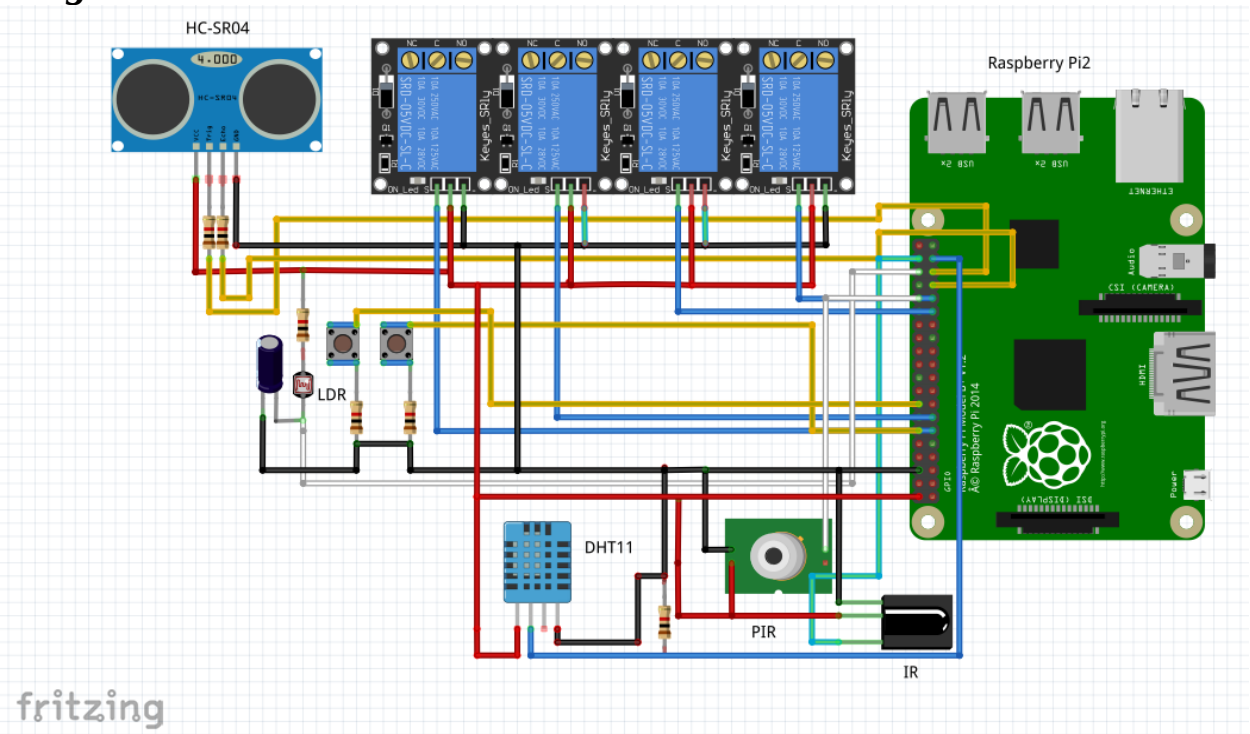


Figure 3-38 Sensors and RPi integration

3.3.4 Webcam, Speaker, Raspberry Pi and Arduino integration

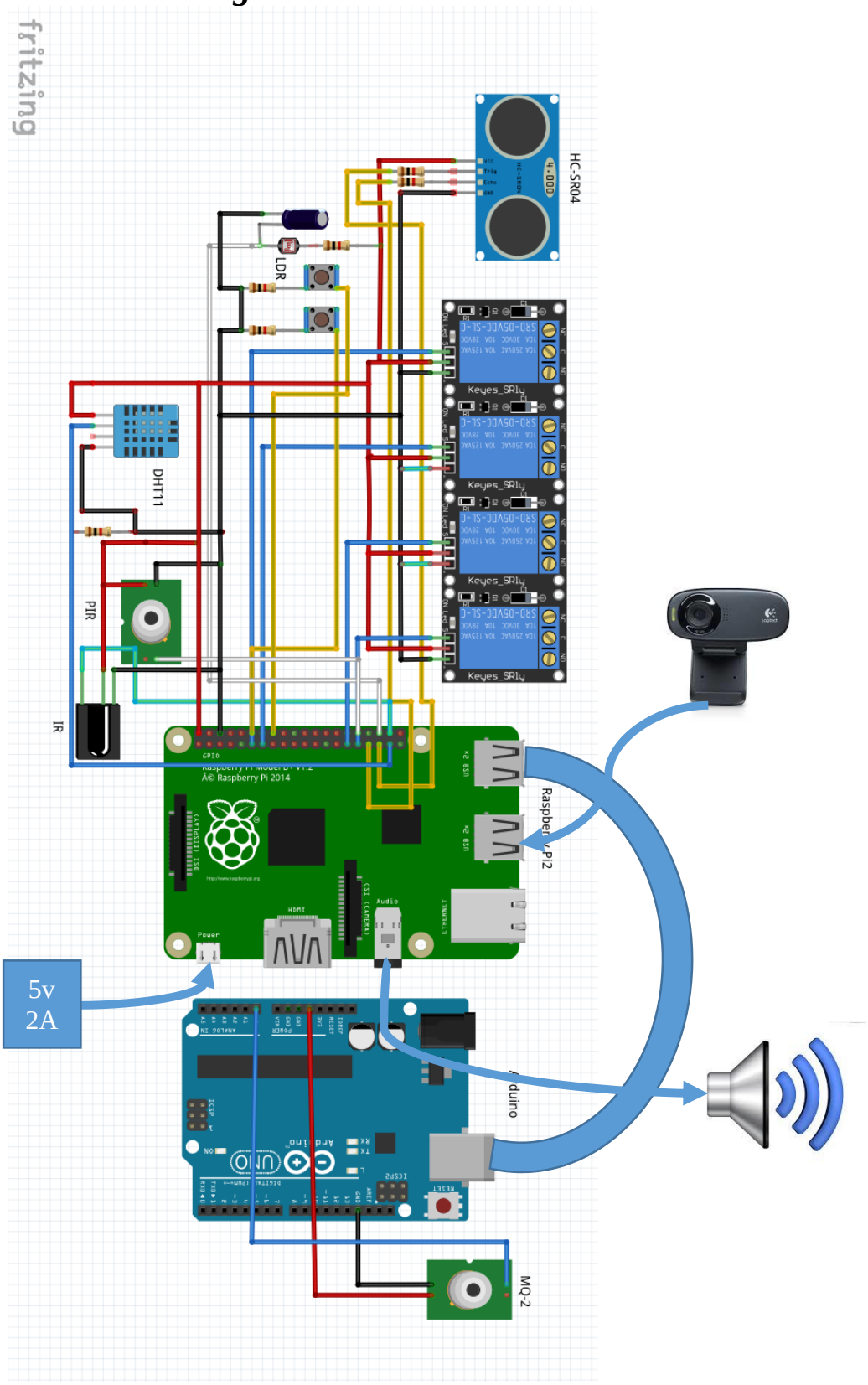


Figure 3-39 RPi and Arduino integration

3.3.5 Final product

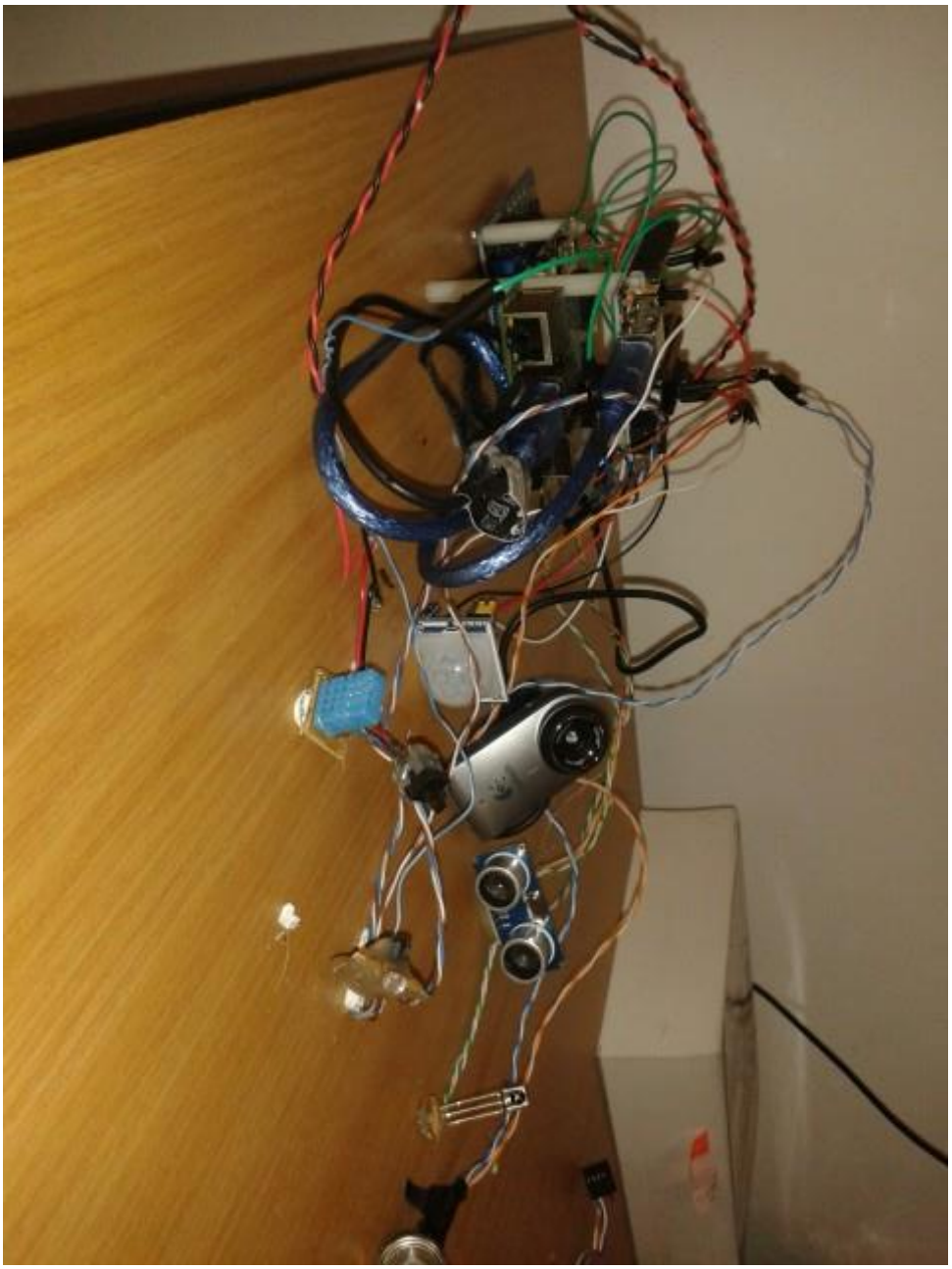


Figure 3-40 Final Product (preliminary)

The final product still requires a package which is to say an enclosure, by the time the image was taken an enclosure hadn't been procured.

3.4 Testing procedure

Due to the modular nature of the design of the proposed project, it was straight forward to test each module individually. In doing so, bugs were easy to spot, locate, and fixed.

3.4.1 Infrared control with RPi

In this section we're going to test an IR sensor and then hook it up to our Raspberry Pi and programme a remote to interact with it!

IR connection to RPi

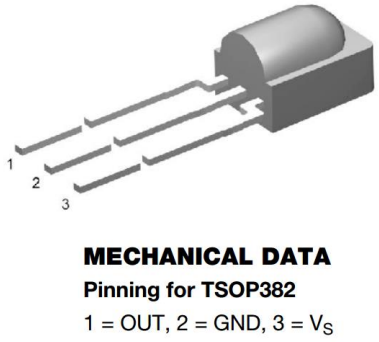


Figure 3-41 IR Receiver

Figure 3-41 IR Receiver shows, an IR receiver pin out connections.
Connect up the sensor like so:

- Pin 1 is the output
- Pin 2 is GND
- Pin 3 is VCC, connect to 5V

The positive (longer) head of the Red LED connects to the +5V pin and the negative (shorter lead) connects through a 200 to 1kohm resistor to the first pin on the IR sensor.

Attaching IR Sensor to the Raspberry Pi

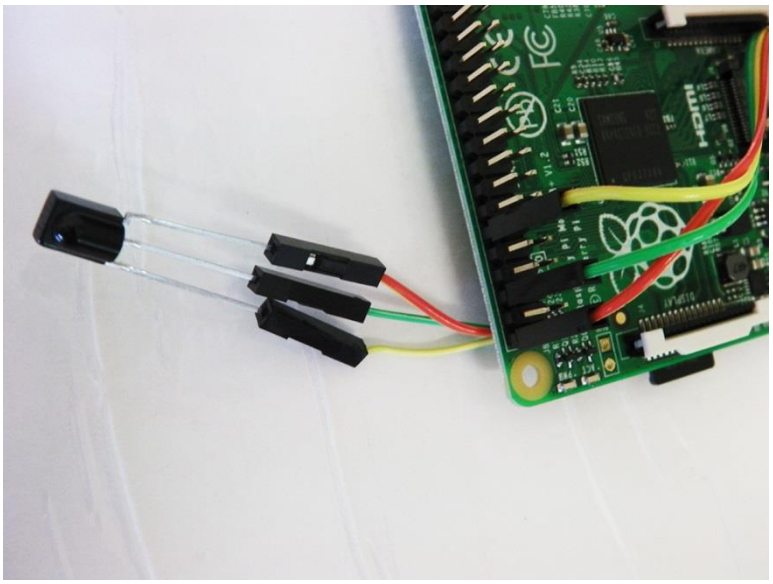


Figure 3-42 IR and RPi interface

Now that we're happy our receiver is working, we're going to hook it up to our Raspberry Pi, and configure it with a remote control.

- Pin 1 is DATA, goes to RPi pin 12 (GPIO 18);
- Pin 2 is GND, goes to RPI pin 6 (GROUND)
- Pin 3 is POWER, goes RPi pin 1 (5V)

Testing the IR Sensor on the Raspberry Pi

To check if the IR Sensor is working on the Pi, we run the following commands one after each other, see output on Figure
sudo modprobe lirc_rpi
sudo kill $(pidof lircd)
mode2 -d /dev/lirc0

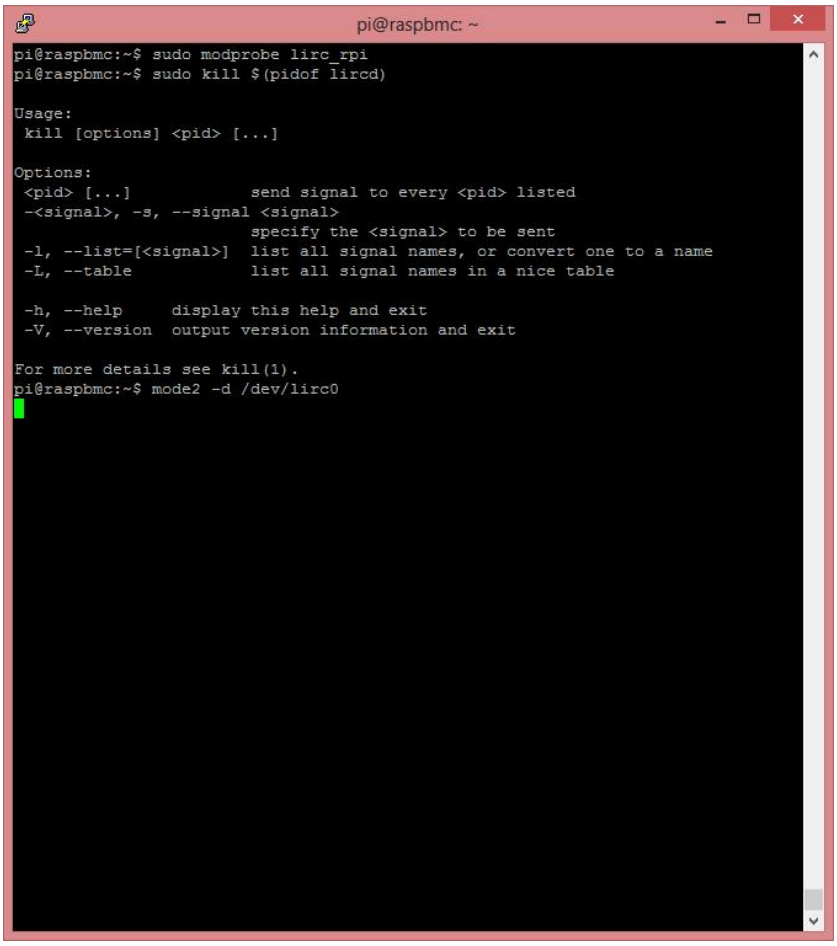


Figure 3-43 IR Commands

Now, when you press buttons on your remote, assuming your receiver is connected correctly, you should see something resembling this appear on the screen with each press, See Figure below.

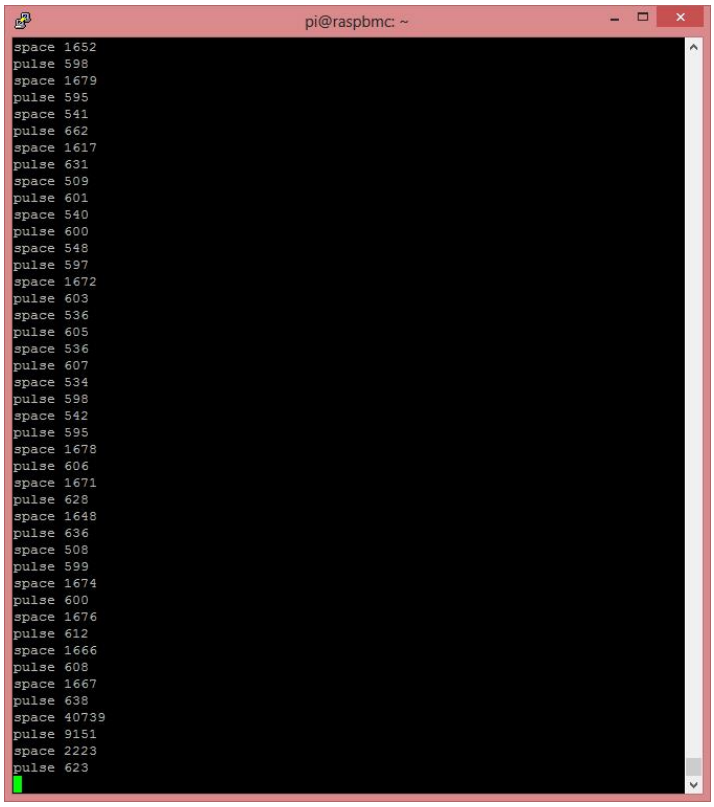


Figure 3-44 IR Test output

Recording your remote

First thing to do is to get the list of KEY commands that are accepted.

To do this, we run the commands

sudo kill $(pidof lircd)
irrecord --list-namespace | grep KEY

Once that is saved to the list we are ready to record the remote.

To record your remote, run the following commands
sudo kill $(pidof lircd)
irrecord -d /dev/lirc0 ~/lircd.conf

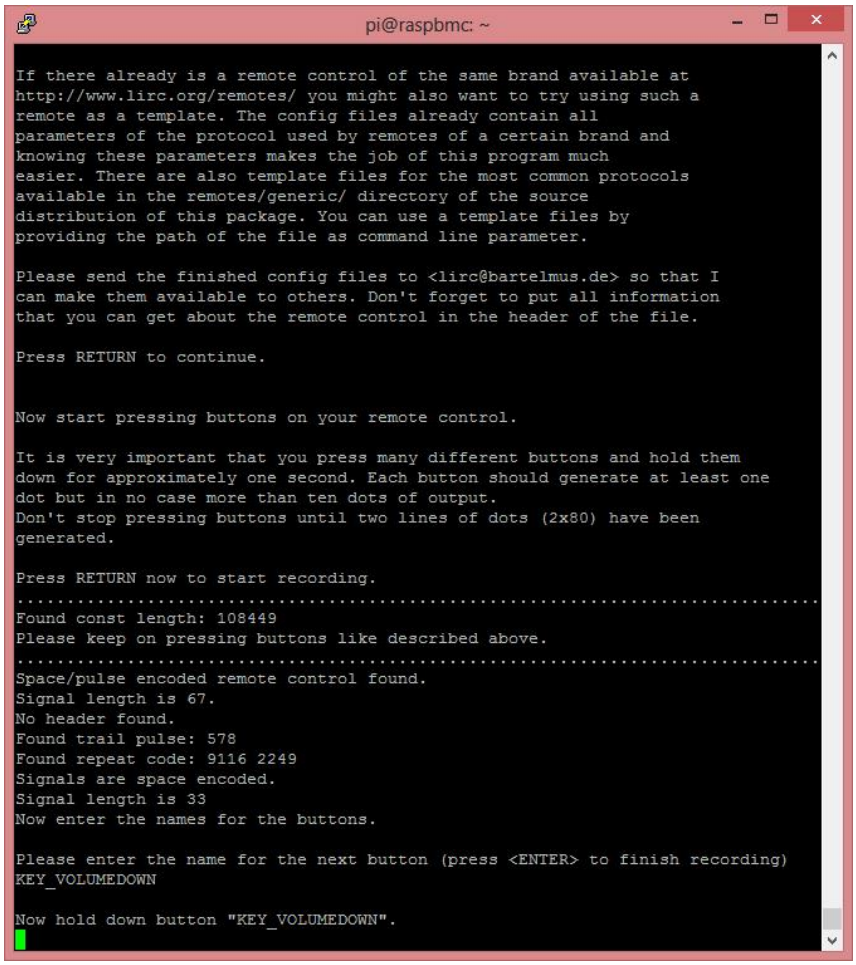


Figure 3-45 Setting up remote}

We follow the instructions given by irrecord exactly.

When you get to the bit where you are asked "Please enter the name for the next button (press <ENTER> to finish recording)"

This is where you start to configure each button.

So we pick a button on the remote we want to configure, for example on our IR Remote Control, the first button is the Volume Down button.

So we need to find the KEY command that we want to associate with the Volume Down button. In this case, it makes sense to use the KEY_VOLUMEDOWN key. You can however configure any KEY command to any button. See Figure

Once you have found the KEY command for the button you want configure, simply enter that KEY and hit ENTER, you should now be prompted to press and hold the button on the remote you want to assign to this KEY.

Now do that for every button you want to use on the remote.

After you have configured the last button, simply hit ENTER to stop recording.

We'll be asked to repeatedly press an arbitrary
button as fast as you can (make sure you press the same button each time, and that you don't hold the button down).

The recording has now been completed, and hopefully our *lircd.conf* has been created (this can be found here: */home/pi/lircd.conf)*

Reconfigure codes to use on Python

```
import lirc
"""
begin
    prog = irexec
    button = KEY_1
    config = echo "You pressed one"
    repeat = 0
end
"""
lirc.init('irexec')

while True:
    btn = lirc.nextcode()
    if btn != []:
        print btn
```

3.4.2 Temperature and Humidity Sensing with RPi

The DHT11 is a basic, ultra low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin (no analog input pins needed). It's fairly simple to use, but requires careful timing to grab data. The only real downside of this sensor is you can only get new data from it once every 2 seconds, so when using our library, sensor readings can be up to 2 seconds old.

Figure 3-12 DHT11 Sensor shows, The DHT11 is a 4-pin (one pin is unused) temperature and humidity sensor capable of measuring 20% - 90% relative humidity and 0 to 50°C. The sensor can operate between 3 and 5.5V DC and communicates using its own proprietary One-Wire protocol.

This protocol requires very precise timing in order to get the data from the sensor. The LOW and HIGH bits are coded on the wire by the length of time the signal is HIGH. The total time to take a reading is at most 23.4ms. This includes a 18ms delay required to start the data transfer and a window of up to 5.4ms for the data. Individual signals can be as short as 20uS and as long as 80uS.

Interfacing DHT11 with Raspberry P

Figure 3-46 DHT11 Pin Connection shows the connections on DHT11, To power the sensor, we connect pin '1' to the 5V terminal of the RPi, pin '2' to voltage divide resistor 4k7 ohms and the Vout will be our input to the RPi GPIO16, pin '3' is unused and pin '4' to the GND terminal of the RPi, This gives the sensor the 5 volts it needs to be powered.

Figure 3-47 DHT11-Wiring shows, the physical connections in order for our circuit to work.

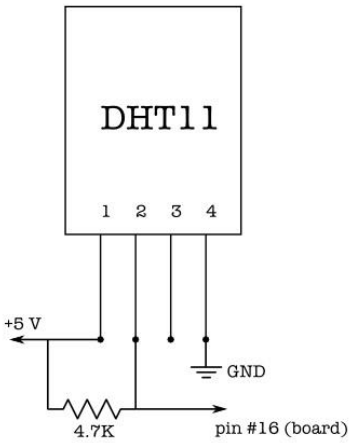


Figure 3-46 DHT11 Pin Connection

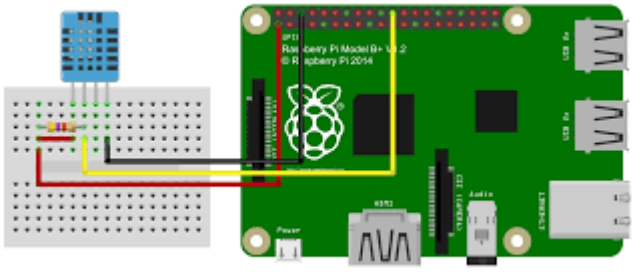


Figure 3-47 DHT11-Wiring

Bash Script Testing

First we will need to grab and install the bcm2835 library before building it. I grabbed the latest sources for Mike McCauley's bcm2835 library and installed them on the pi. [15]

```
pi@raspberrypi ~ $ wget http://www.open.com.au/mikem/bcm2835/bcm2835-1.14.tar.gz
pi@raspberrypi ~ $ tar xvzf bcm2835-1.14.tar.gz
pi@raspberrypi ~ $ cd bcm2835-1.14
pi@raspberrypi ~/bcm2835-1.14 $ ./configure
pi@raspberrypi ~/bcm2835-1.14 $ make
pi@raspberrypi ~/bcm2835-1.14 $ sudo make install
```

Download the software, you will need git so install that first.

pi@raspberrypi ~ $ apt-get install git
pi@raspberrypi ~ $ git clone https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code.git
pi@raspberrypi ~ $ cd Adafruit-Raspberry-Pi-Python-Code/
pi@raspberrypi ~ $~/Adafruit-Raspberry-Pi-Python-Code

Build the software.
pi@raspberrypi ~ $ cd ~/Adafruit-Raspberry-Pi-Python-Code/Adafruit_DHT_Driver $ make

Run the Adafruit_DHT command.
pi@raspberrypi ~ $ cd ~/Adafruit-Raspberry-Pi-Python-Code/Adafruit_DHT_Driver
pi@raspberrypi ~ $ sudo ./Adafruit_DHT 2302 4
Adafruit_DHT 2302 4
Using pin #4
Data (40): 0x2 0x3e 0x0 0xde 0x1e
Temp = 22.2 *C, Hum = 57.4 %

3.4.3 Ultrasonic Distance Sensors with RPi

Sound consists of oscillating waves through a medium (such as air) with the pitch being determined by the closeness of those waves to each other, defined as the frequency. Only some of the sound spectrum (the range of sound wave frequencies) is audible to the human ear, defined as the "Acoustic" range. Very low frequency sound below Acoustic is defined as "Infrasound", with high frequency sounds above, called "Ultrasound". Ultrasonic sensors are designed to sense object proximity or range using ultrasound reflection, similar to radar, to calculate the time it takes to reflect ultrasound waves between the sensor and a solid object. Ultrasound is mainly used because it's inaudible to the human ear and is relatively accurate within short distances. You could of course use Acoustic sound for this purpose, but you would have a noisy robot, beeping every few seconds.

Figure 3-48 HC-SR04 Tx/Rx shows a basic ultrasonic sensor consists of one or more ultrasonic transmitters (basically speakers), a receiver, and a control circuit. The transmitters emit a high frequency ultrasonic sound, which bounce off any nearby solid objects.

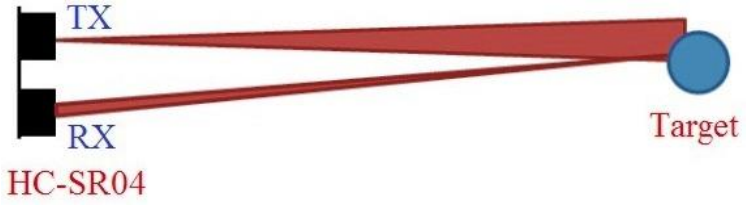


Figure 3-48 HC-SR04 Tx/Rx

Some of that ultrasonic noise is reflected and detected by the receiver on the sensor. That return signal is then processed by the control circuit to calculate the time difference between the signal being transmitted and received. This time can subsequently be used, along with some clever math, to calculate the distance between the sensor and the reflecting object.

The HC-SR04 Ultrasonic sensor we'll be using has four pins: ground (GND), Echo Pulse Output (ECHO), Trigger Pulse Input (TRIG), and 5V Supply (Vcc). See Figure 3-32 HC-SR04

We power the module using Vcc, ground it using GND, and use our Raspberry Pi to send an input signal to TRIG, which triggers the sensor to send an ultrasonic pulse. The pulse waves bounce off any nearby objects and some are reflected back to the sensor. The sensor detects these return waves

and measures the time between the trigger and
returned pulse, and then sends a 5V signal on the ECHO pin.

ECHO will be “low” (0V) until the sensor is triggered when it receives the echo pulse. Once a return pulse has been located ECHO is set “high” (5V) for the duration of that pulse. Pulse duration is the full time between the sensor outputting an ultrasonic pulse, and the return pulse being detected by the sensor receiver. Our Python script must therefore measure the pulse duration and then calculate distance from this.

IMPORTANT. The sensor output signal (ECHO) on the HC-SR04 is rated at 5V. However, the input pin on the Raspberry Pi GPIO is rated at 3.3V. Sending a 5V signal into that unprotected 3.3V input port could damage your GPIO pins, which is something we want to avoid! We’ll need to use a small voltage divider circuit, consisting of two resistors, to lower the sensor output voltage to something our Raspberry Pi can handle.

Voltage Dividers

A voltage divider consists of two resistors (R1 and R2) in series connected to an input voltage (Vin), which needs to be reduced to our output voltage (Vout). In our circuit, Vin will be ECHO, which needs to be decreased from 5V to our Vout of 3.3V.
The voltage divider equation is:

$$\frac{Vo}{Vi} = \frac{R2}{R1 + R2}$$

We will use a 1k ohm for R1 and a 2k ohm resistor as R2 to get a Vout of 3V3 with Vin 5V!

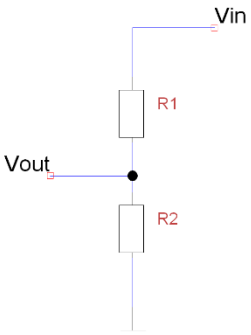


Figure 3-49 Voltage Divider Circuit

Assemble the Circuit

We’ll be using four pins on the Raspberry Pi for this project:
GPIO 5V [Pin 2], Vcc (5V Power), GPIO GND [Pin 6], GND (0V Ground), GPIO 23 [Pin 16], TRIG (GPIO Output) and GPIO 24 [Pin 18], ECHO (GPIO Input)

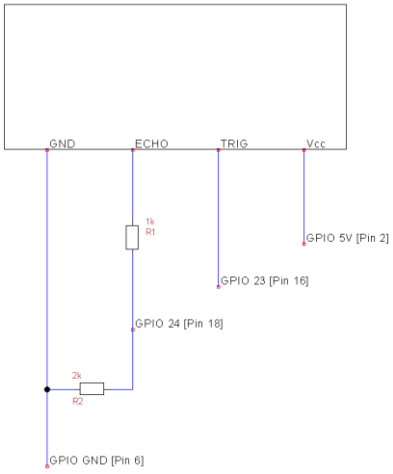


Figure 3-50 HC-SR04 Connections

Figure 3-51 HC-SR04 and RPi Connection
below shows the completed connection between HC-SR04 sensor and Raspberry Pi!

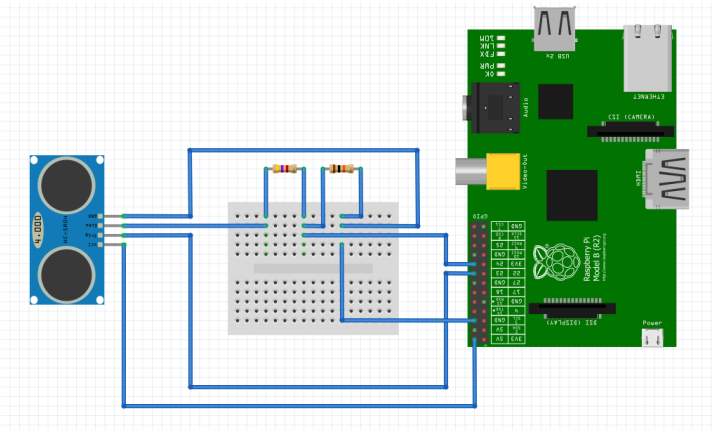


Figure 3-51 HC-SR04 and RPi Connection

Sensing with Python

Now that we have connected our Ultrasonic Sensor up to our RPi, we need to program a Python script to detect distance!
The Ultrasonic sensor output (ECHO) will always output low (0V) unless it's been triggered in which case it will output 5V (3.3V with our voltage divider!). We therefore need to set one GPIO pin as an output, to trigger the sensor, and one as an input to detect the ECHO voltage change.

Distance Calculation

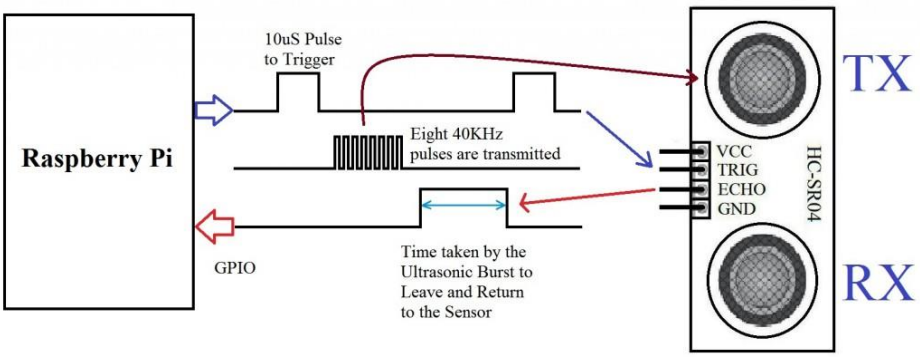


Figure 3-52 Timing RPi and HC-SR04

Time taken by pulse is actually for to and fro travel of ultrasonic signals, while we need only half of this.
Therefore time is taken as Time/2.
*Distance = Speed * Time/2*
Speed of sound at sea level = 343 m/s or 34300 cm/s
*Thus, Distance = 17150 * Time (unit cm)*

Python Programming

```
import RPi.GPIO as GPIO                 #Import GPIO library
import time                           #Import time library
GPIO.setmode(GPIO.BCM)                  #Set GPIO pin numbering
TRIG = 23                             #Associate pin 23 to TRIG
ECHO = 24                             #Associate pin 24 to ECHO

print "Distance measurement in progress"

GPIO.setup(TRIG,GPIO.OUT)               #Set pin as GPIO out
GPIO.setup(ECHO,GPIO.IN)                #Set pin as GPIO in

while True:

  GPIO.output(TRIG, False)              #Set TRIG as LOW
  print "Waiting For Sensor To Settle"
  time.sleep(2)                       #Delay of 2 seconds

  GPIO.output(TRIG, True)               #Set TRIG as HIGH
  time.sleep(0.00001)                 #Delay of 0.00001 seconds
  GPIO.output(TRIG, False)              #Set TRIG as LOW

  while GPIO.input(ECHO)==0:            #Check whether the ECHO is LOW
    pulse_start = time.time()         #Saves the last known time of LOW pulse

  while GPIO.input(ECHO)==1:            #Check whether the ECHO is HIGH
    pulse_end = time.time()           #Saves the last known time of HIGH pulse

  pulse_duration = pulse_end - pulse_start #Get pulse duration to a variable

  distance = pulse_duration * 17150       #Multiply pulse duration by 17150 to get distance
  distance = round(distance, 2)         #Round to two decimal points
  print "Distance:",distance,"cm"         #Print distance in cm calibration
  GPIO.cleanup()
```

```
pi@raspberrypi ~ $ sudo python range_sensor.py
Distance Measurement In Progress
Waiting For Sensor To Settle
Distance: 12.52 cm
pi@raspberrypi ~ $
```

Figure 3-53 SR-HC04 Readings

3.4.4 MQ-2 Smoke Sensor connected to an Arduino

In this section, we will go over how to build a smoke sensor circuit with an Arduino board.
The smoke sensor we use is the MQ-2. This is sensor is not only sensitive to smoke, but also to flammable gas.
The MQ-2 smoke sensor reports smoke by the voltage level that it outputs. The more smoke there is, the greater the voltage that it outputs. Conversely, the less smoke that it is exposed to, the less voltage it outputs.
We will wire the MQ-2 to an Arduino so that the Arduino can read the amount of voltage output by the sensor to serial output.

The code that we need to upload is shown below.

```
/*
 Testing MQ-2 GAS sensor with serial monitor
*/
// Analog pin 0 will be called 'sensor'
const int sensorPin = A0;
// Set the initial sensorValue to 0
int sensorValue = 0;

// The setup routine runs once when you press reset
void setup() {
 // Initialize serial communication at 9600 bits per second
  Serial.begin(9600);
}

// The loop routine runs over and over again forever
void loop() {
  // Read the input on analog pin 0 (named 'sensor')
  sensorValue = analogRead(sensor);
  // Print out the value you read
  Serial.println(sensorValue);
  delay(1000); // Print value every 1 sec.
}
```

The first block of code declares and initialises 2 variables. The sensorPin represents the smoke sensor. The sensorValue is initialized to 0, because it will be connected to analog pin A0 of the Arduino board.
The next block of code defines the baud rate and the input and output of the circuit. The sensorPin, which is the smoke sensor pin, serves as the input of the circuit. This sensor is input into the Arduino so that the Arduino can read and process the value.
The next block of code uses the analogRead() function to read the value from the sensorPin (the smoke sensor). This will be a numerical value from 0 to 1023. 0 represents no smoke, while 1023 represents smoke at the absolute maximum highest level. So the variable, sensorValue, represents the smoke level that can range from 0 to 1023. We put a line to print this value just for debugging purposes, so that you can see what values are being returned from this function.
This last block of code was the loop() function. This is the part of code that repeats over and over in an infinite loop with 1 second delay. This means that our code is always checking to see what the sensorValue is.
And this is how a smoke sensor works with an MQ-2 and an Arduino.

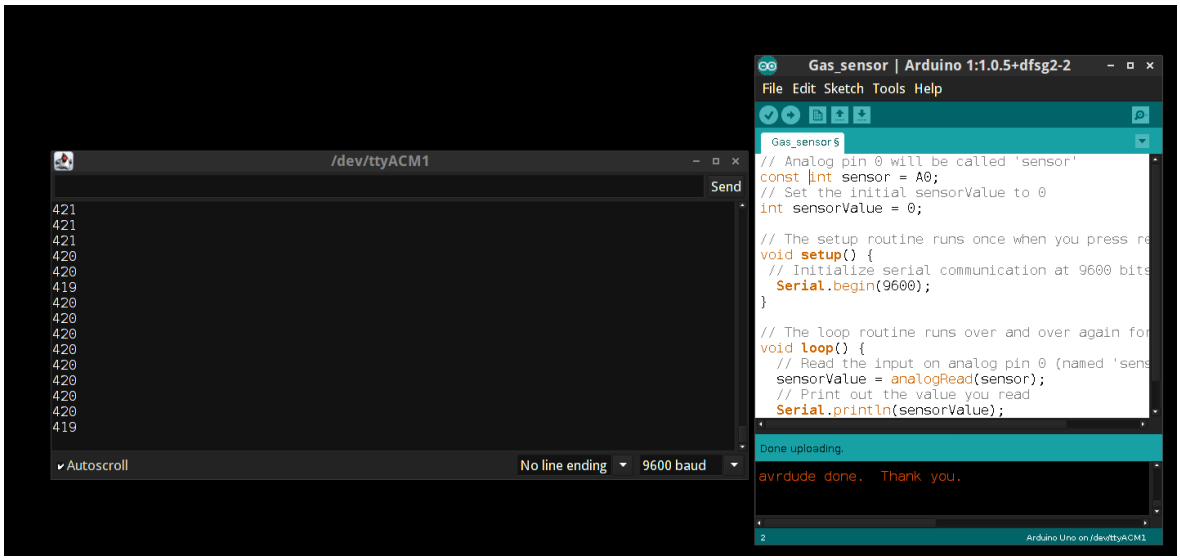


Figure 3-54 Arduino IDE with serial output

Arduino MQ-2 Smoke Sensor Circuit Schematic

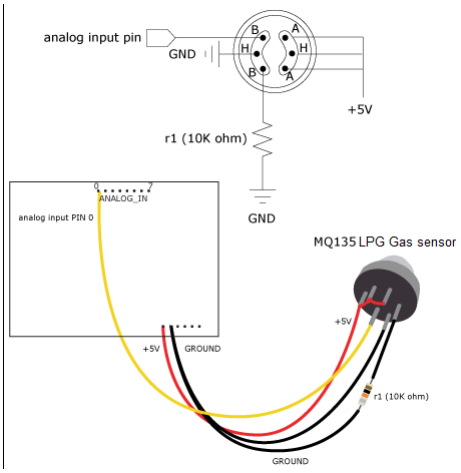


Figure 3-55 MQ2 connections

Figure 3-55 MQ2 connections shows the connections, To power the smoke sensor, we connect pin 'A' and 'H' of the smoke sensor to the 5V terminal of the Arduino and pin 'H' to the GND terminal of the Arduino, pin 'B' to 10k ohm resistor then to the GND terminal of the Arduino.
This gives the smoke sensor the 5 volts it needs to be powered.

The output pin 'B' of the sensor goes into analog pin A0 of the Arduino. Through this connection, the Arduino can read the analog voltage output from the sensor. The Arduino board has a built-in analog-to-digital converter, so it is able to read analog values without any external ADC chip.

Depending on the value that the Arduino reads determines the action that will occur with the circuit. Our current setup is just to read real-time values being received by the sensor.

Figure 3-56 Arduino-MQ2 testing shows, the physical connections in order for our circuit to work.

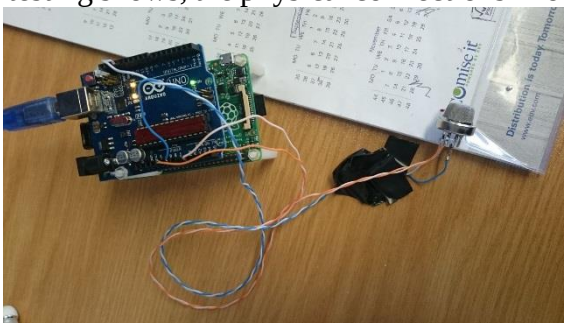


Figure 3-56 Arduino-MQ2 testing

3.4.5 Necessary changes encountered

Since the project inception there has been various form of changes, mainly on the software architecture and also in terms of integration between various features of the proposed project.
Initially the relay modules where to be ran on the Arduino instead of the Raspberry Pi due to the limited GPIO pins the Raspberry Pi has, the reason behind this change was due to the Arduino serial CTS(Clear to send) which was resetting the GPIO pins of the Arduino.
The Raspberry PI's GPIO are configured differently, most are configured as inputs and others as outputs upon boot by default, and this led to some components of the project to be moved around from input pin to output pin.

The API used for plotting the sensor data was changed from Plotly to Thinkspeak as Plotly was a pay per use service as compared to Thinkspeak being an open source service.

Most of the software requires continuous integration and package updates, for instance Google's text to speech API encountered a lot of changes due to the security feature Google has implemented by making use of Captcha's to access their API's. A Captcha is a computer program or system intended to distinguish human from machine input, typically as a way of preventing spam and automated extraction of data from websites.

The Smart Doorbell feature initially had to be configured with a feature to make a Telepresense possible. Telepresense refers to the application of complex video technologies to give geographically separated participants a sense of being together in the same location. [14]
But due to the network latency, the was a 2 seconds lag time, which mean the video would not be in real-time and the resources of the proposed project where over used when this feature was enable. This led to the feature to be deprecated.

3.5 Conclusion

The project was designed according to the specifications and testing procedures have been developed to test whether actual results are favourable with regards to the required result. With regards to this chapter, we have been able to look at different stage to accomplish the project. The next chapter will deal with the result and comparison with the actual and required result.

4 Chapter 4

4.1 Introduction

This chapter shows and discusses, results of the working model or product that is this section will cover the measurements done during testing and tested procedure performed and analyzed during the implementation of this product. A comparison will be drawn from the requirement on the project and what has been done in the project

4.2 Results of the tested product / procedure

Table 4-1 Results of the tested product

Product	Requirement
Raspberry Pi	The prime use of the Raspberry Pi is to control all features of the proposed project. The Raspberry Pi is the brain of the project and it is where 70% of the project including software was written with the main programming language being Python and C
Arduino Uno	The Arduino Uno is currently being used as an ADC with the MQ2 gas sensor connected to it. Future plans to utilize its GPIO pins and additional processing power are in planning phase
Integrated Modules	Relays: As per 3.3.1, By using relays were where able to control appliances and lightings that use AC supplies, instead of directly connecting them to the Raspberry Pi – which would never be advisable. LDR: By means of connecting the LDR directly to the Raspberry Pi without wasting any ADC pins on the Arduino Uno we were successful into reading light intensity levels and upload them to an online graphing website Speakers: By means of having speakers connected to the Raspberry Pi we were able to listen to the output from and feedback/notifications from the various features such as the Smart Doorbell and Smart Alarm Webcam: However not being able to stream real time video via UDP protocol, we were able to use the webcam as real time security surveillance and smart doorbell notification by capturing an image of the visitor and sending it to the user as explained in chapter 3 IR: By using the IR receiver we were able to control actions on the

	Raspberry Pi, such as switching on/off appliances(HVAC/Heater and Fan) and lightings Smoke Sensor: With the smoke sensor connected to the Arduino and having the Arduino connected to the Raspberry Pi via serial we were able to retrieve smoke levels from the sensor with push notification and cloud data upload implemented or included. PIR: Having a PIR on the project meant that we are able to have an additional feature in the form of security.

4.3 Comparison of results vs. requirements

In this section we will compare the requirements mentioned in paragraph and results

Table 4-2 Comparison of results vs. requirements

Product	**Results**	**Comparison**
Raspberry Pi	Worked as expected	Similar
Arduino Uno	Worked as expected	Similar
Relays	Worked as expected	Similar
LDR	Worked as expected	Similar
Speakers	Worked as expected	Similar
Webcam	It was discovered that real-time video streaming was not achievable due to various issues arising from network latency and resource management as it increased the amount of CPU and Memory usages.	Different, an option of using static images instead of video was implemented
IR remote	Worked as expected	Similar
Smoke sensor	Worked as expected	Similar
PIR	Worked as expected	Similar
Smart Doorbell	Due to API's used changes were made to the feature of the project and it requires constant package updates	Due to API updates, software requires occasional updates
Ultrasonic sensor	Worked as expected	Similar
Temperature and	It was realized after noticing that	Results are the same however,

humidity sensor	the Python script requires more resources and it is also slow when retrieving data, a C code to read temperature and humidity values was written as it offers low level communication and it is also fast	resources are managed better
Laptop	Worked as expected	Website control offered reliable results and does not require updates
Android based-phone	Android application needs improvements as the URL is static instead of being dynamic	Results are similar, with a simple option of dynamically inputting the URL

.

4.4 Conclusion

The comparison between the task required and the result on the project was discussed. We can conclude that the project has met the requirement as we have discussed the results obtained vs requirements.

5 Chapter 5

5.1 Introduction

This chapter evaluates the success of this project and gives some recommendations for improvements and conclusions. The overview of the entire project and summary will be discussed here. The challenges and the duration taken for the completion will also be mentioned likewise the final financial budget compared to the proposed budget and a proposed further study.

5.2 Conclusions and recommendations

It is evident from this project work that an individual control home automation system can be cheaply made from low-cost locally available components and can be used to control miscellaneous home appliances ranging from the security lightings, television control, air conditioning system, security surveillance and even the entire house lighting system.
The components required are small and few that they can be packaged into a small inconspicuous enclosure considering it has air flow capabilities. The proposed system was tested a numerous amount of times and certified to control different home appliances used in the lighting system, air conditioning system, heating system, home entertainment system and many more (this is as long as the maximum power and current rating of the appliance does not exceed that of the used relay which is 220V 10A). Finally, this system can be also implemented over Wi-Fi or LAN connectivity without much change to the design and yet still be able to control a variety of home appliances. Hence, this system is scalable and flexible.

In conclusion, we feel that our product is completely sound, and has great market value.
The advantage of our product is that there are no known competitors, as it has its own unique features. Secondly, our project allows for greater development of products. The code is open source and we are providing for development by others for application in several ways.
With our software, we could develop it for more complicated applications such as sophisticated home surveillance and security with geolocation and monitoring, pet tracker or kid's tracker. In particular, we will provide installation instructions on setting up the product. Even though we were unable to fully complete most of the features of the project in time, we have the system almost complete and ready. Hence we feel that there is no better product available in the market.

5.3 Financial cost and time evaluation

Table 5-1 Overall Budget Evaluation

Item	Description	Quantity	Unit Estimated Price (ZAR)	Unit Actual Price (ZAR)	Source	Total Price (ZAR)
1	Raspberry Pi B+	1	850.00	741	communica	741.00
2	Arduino Uno R3	1	190.00	307	communica	190
3	USB Webcam	1	120.00	125	communica	125.00
4	USB Wi-Fi Dongle	1	100.00	189	communica	189.00
5	USB Speakers	1	50.00	79.90	communica	79.90
6	5V 2A Power supply	1	150.00	150	communica	150
7	Relay modules	2	250.00	129	communica	258
8	Enclosure	1	200.00	329	communica	329.00
9	Prototype Board	1	10.00	75	communica	75.00
10	Pack of various Resistors	100	200	355	communica	355
11	1uF 16V capacitor	4		.66	communica	2.64
12	Push Button	2		33	communica	66
13	DHT11	1	800.00	85	communica	85.00
14	HC-SR04	1		85	communica	85.00
15	PIR	1		75	communica	75.00
16	LDR	2		1.50	communica	3.00
17	MQ2	1		49	communica	49.00
18	IR Rx	1		65	communica	65.00
	Total Cost		R2920			R 2922.40

By visually comparing the estimated cost of the overall project cost evaluation one can see that the estimated prices do not differ much from the estimated, which means that the project is cost efficient and mass production would not cost large amounts of money.

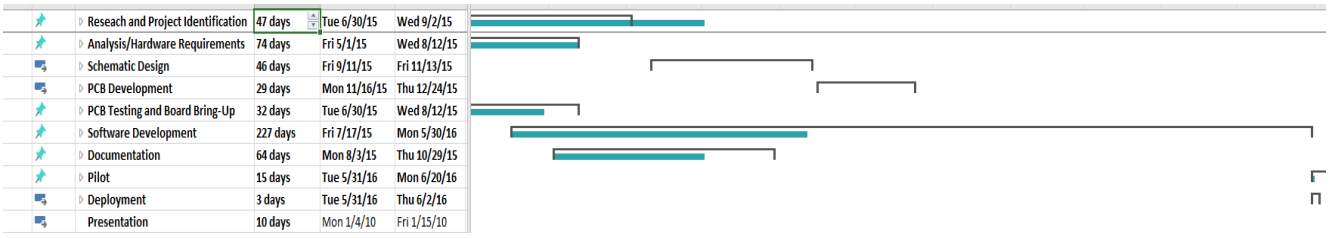


Figure 5-1 Brief Gantt chart

Initially the project report was compiled using **LATEX,** due to unforeseen circumstances I had to redo the almost complete project documentation using the Microsoft Word template issued by the university.

Other factors include continuous improvement on the software architecture which took more time than expected, as well as integration.

5.4 Proposed further study

5.4.1 Natural Language Processing: Speech Recognition

Natural language processing (NLP) is a method to translate between computer and human languages. It is a method of getting a computer to understandably read a line of text without the computer being fed some sort of clue or calculation. In other words, NLP automates the translation process between computers and humans.

5.4.2 Smart Pet Feeder

With the smart pet feeder, you can monitor your pet's eating habits, remotely schedule feedings, and dispense perfectly portioned meals with the app.

5.4.3 Smart Wardrobe

People nowadays like to shopping and buy clothes. However most of the clothes will be stored inside wardrobe for long time even up to several years. Clothes can easily get moulded if placed aside for long time.

Smart wardrobe will help them to manage their clothes inside wardrobe. It will also push all data into Google Drive and further sync up data with Android mobile application. User(s) can get suggestion on what to wear today from their phone app based on what event are there in their Google calendar and also the weather via Openweather's API.

The application will provide a dashboard to show which cloth has the highest frequency of being worn and which has not been worn over long periods of time. Application can further suggest users to sell it into second hand platform or make a donation for charity.

5.4.4 Android Geolocation Detector

A system whereby Lights or plugs can be switched on or off remotely via geolocation. For instance, Imagine you driving home in the middle of the night and considering that the might not be anyone in the house at that time. As soon as Google Maps API detects that you are on a 5km radius of your home, the phone sends a signal to your home automation system which then switches on selected lights, TV and etc.

5.4.5 Light Alarm Clock

Instead of having being woken up by a dreaded ringing loud noise, this alarm system would gradually increase its light intensity until it is bright enough in one's room, simulating a sunrise. Together integrated with ambient noise this makes the waking up experience natural.

Appendix

5.5 A.1 Final Gantt chart

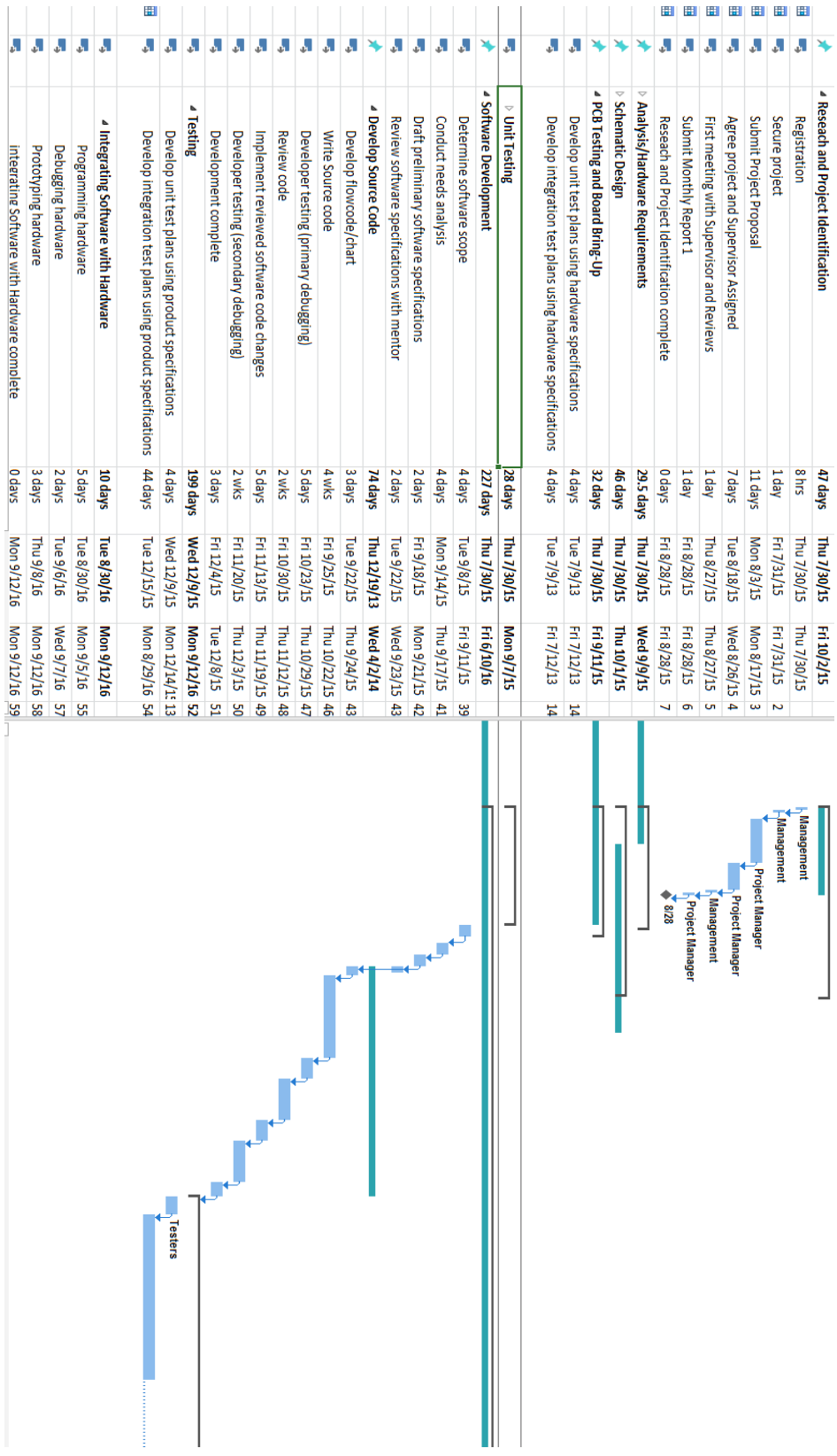


Figure 5-2 Detailed Gantt chart

https://drive.google.com/open?id=0B0gORe2m_qleM1F3bE4wbTJJSlk

5.6 A.2 Bibliography

[1] Android Based Home Automation Using Raspberry Pi

http://www.ijcat.org/IJCAT-2014/1-1/Android-Based-Home-Automation-Using-Raspberry-Pi.pdf

Accessed on [2016-02-09]

[2] HomeAutomation

http://home.hit.no/~hansha/documents/theses/2015/Home%20Automation/home_automation_2015_report.pdf

Accessed on [2016-02-15]

[3] Qwik Switch

www.lightingwarehouse.co.za/Articles/LW-QwikSwitch-2012-02.pdf

Accessed on [2015-11-27]

[4] Scripting layer for Android

https://en.wikipedia.org/wiki/Scripting_Layer_for_Android

Accessed on [2014-12-07]

[5] Accelerometer

https://en.wikipedia.org/wiki/Accelerometer

Accessed on [2016-04-12]

[6] Low pass filter

https://en.wikipedia.org/wiki/Low-pass_filter

Accessed on [2016-04-16]

[7] LIRC

http://www.lirc.org/

Accessed on [2016-04-16]

[8] RC Circuit

https://en.wikipedia.org/wiki/RC_circuit

Accessed on [2016-04-30]

[9] LDR datasheet

http://www.gotronic.fr/pj-1284.pdf

Accessed on [2016-04-17]

[10] MAC address

https://en.wikipedia.org/wiki/MAC_address

Accessed on [2016-04-17]

[11] Use arp-scan to find hidden devices in your network

https://www.blackmoreops.com/2015/12/31/use-arp-scan-to-find-hidden-devices-in-your-network/

Accessed on [2016-04-26]

[12] MQ Gas Sensors

http://playground.arduino.cc/Main/MQGasSensors

Accessed on [2016-02-26]

[13] Ultrasonic Transducer

https://en.wikipedia.org/wiki/Ultrasonic_transducer

Accessed on [2016-05-05]

[14] What is Telepresense?

https://net.educause.edu/ir/library/pdf/ELI7053.pdf

Accessed on [2016-05-10]

[15] C library for Broadcom BCM 2835 as used in Raspberry Pi

http://www.airspayce.com/mikem/bcm2835/

Accessed on [2016-02-12]

5.7 A.3 Detail designs

Schematic:

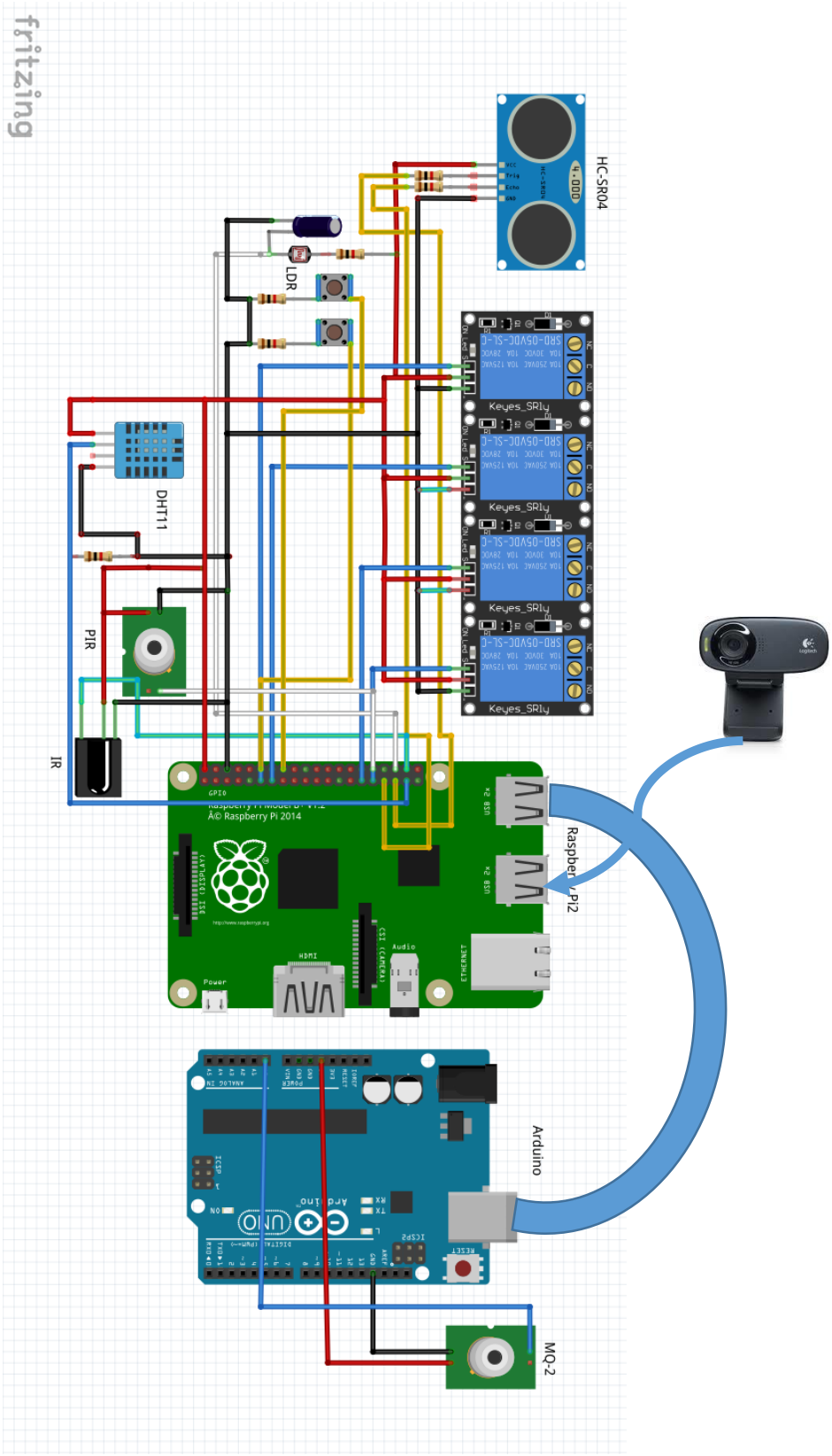


Figure 5-3 Detail Schematic

Component list:

1x Raspberry Pi B+

1x Arduino Uno R3

1x USB-A to USB-B cable

1x USB-A to USB-Micro Cable

1x 8 GB Micro SD Card

1x Nano USB Wi-Fi dongle

2x 4x 5V Relay Modules

1x PIR

1x IR Rx

1x HC-SR04

1x MQ2

1x USB Webcam

2x Push Button switch

1x 1uF 16V Electrolyte Capacitor

1x DHT11

1x USB Speakers

10x 1k Resistors

5.8 A.4 Software

All software (open source) can be accessed from Github:

https://github.com/mmphego/Home-Auto-Pi

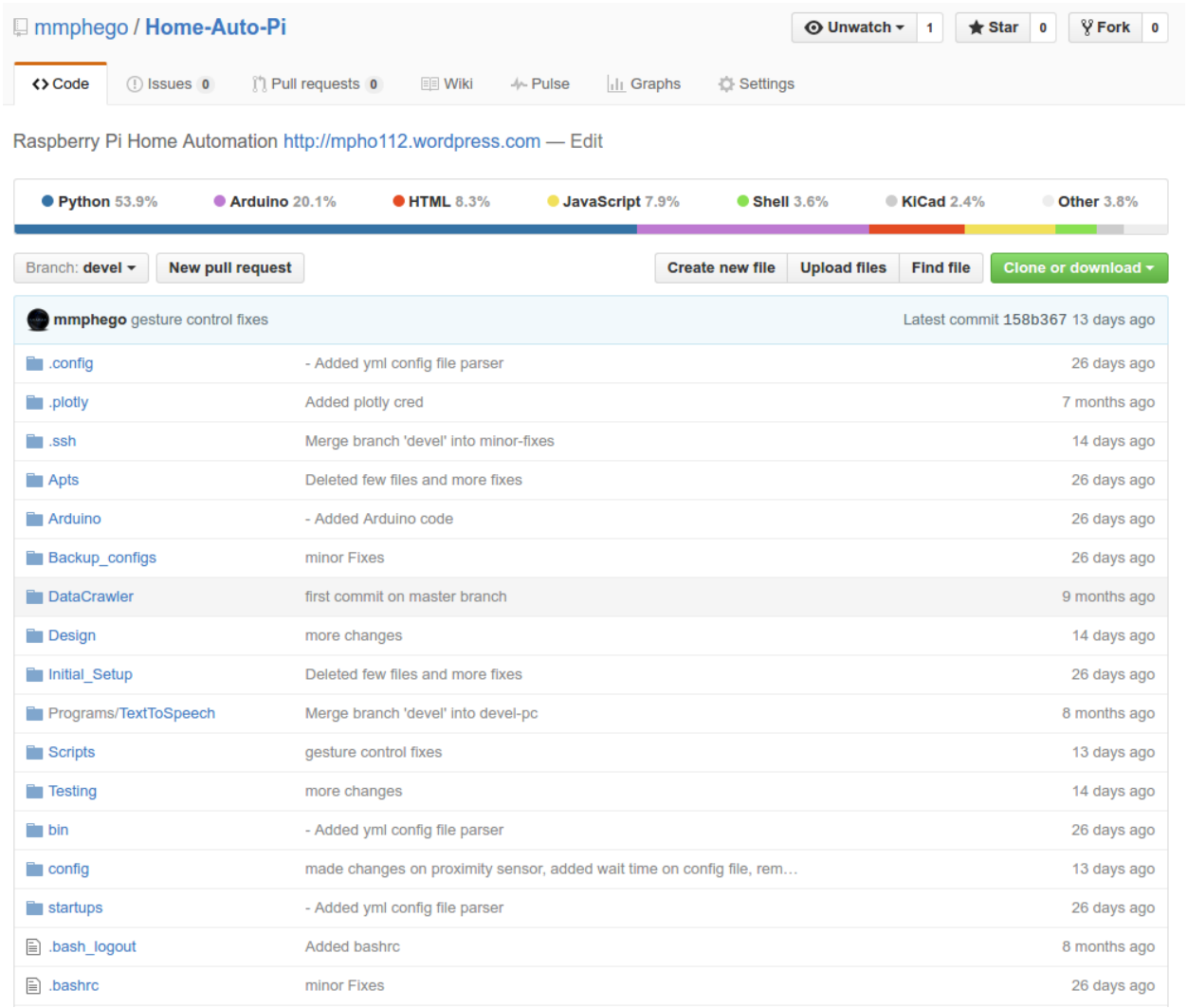


Figure 5-4 GitHub Language Frequency

5.9A.5 Datasheets

Overview
Technical Specs
Documentation

Technical specs

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Relative humidity
Resolution: 16Bit
Repeatability: ±1% RH
Accuracy: At 25℃ ±5% RH
Interchangeability: fully interchangeable
Response time: 1 / e (63%) of 25℃ 6s
1m / s air 6s
Hysteresis: <± 0.3% RH
Long-term stability: <± 0.5% RH / yr in

Temperature
Resolution: 16Bit
Repeatability: ±0.2℃
Range: At 25℃ ±2℃
Response time: 1 / e (63%) 10S

Electrical Characteristics
Power supply: DC 3.5 ~ 5.5V
Supply Current: measurement 0.3mA standby 60μ A
Sampling period: more than 2 seconds

Pin Description
1, the VDD power supply 3.5 ~ 5.5V DC
2 DATA serial data, a single bus
3, NC, empty pin
4, GND ground, the negative power

EVERLIGHT

Technical Data Sheet

5mm Infrared LED , T-1 3/4

IR333-A

Features

- High reliability
- High radiant intensity
- Peak wavelength λ p=940nm
- 2.54mm Lead spacing
- Low forward voltage
- Pb free
- The product itself will remain within RoHS compliant version.

Descriptions

- EVERLIGHT'S Infrared Emitting Diode(IR333-A) is a high intensity diode , molded in a blue transparent plastic package.
- The device is spectrally matched with phototransistor , photodiode and infrared receiver module.

Applications

- Free air transmission system
- Infrared remote control units with high power requirement
- Smoke detector
- Infrared applied system

Device Selection Guide

LED Part No.	Chip Material	Lens Color
IR	GaAlAs	Blue

HANWEI ELETRONICS CO.,LTD MQ-2 http://www.hwsensor.com

TECHNICAL DATA MQ-2 GAS SENSOR

FEATURES

Wide detecting scope — Fast response and High sensitivity
Stable and long life — Simple drive circuit

APPLICATION

They are used in gas leakage detecting equipments in family and industry, are suitable for detecting of LPG, i-butane, propane, methane ,alcohol, Hydrogen, smoke.

SPECIFICATIONS

A. Standard work condition

Symbol	Parameter name	Technical condition	Remarks
Vc	Circuit voltage	5V±0.1	AC OR DC
V_H	Heating voltage	5V±0.1	ACOR DC
R_L	Load resistance	can adjust	
R_H	Heater resistance	33Ω ±5%	Room Tem
P_H	Heating consumption	less than 800mw	

B. Environment condition

Symbol	Parameter name	Technical condition	Remarks
Tao	Using Tem	-20℃-50℃	
Tas	Storage Tem	-20℃-70℃	
R_H	Related humidity	less than 95%Rh	
O_2	Oxygen concentration	21%(standard condition)Oxygen concentration can affect sensitivity	minimum value is over 2%

C. Sensitivity characteristic

Symbol	Parameter name	Technical parameter	Remarks
Rs	Sensing Resistance	3KΩ-30KΩ (1000ppm iso-butane)	Detecting concentration scope: 200ppm-5000ppm LPG and propane 300ppm-5000ppm butane 5000ppm-20000ppm methane 300ppm-5000ppm H_2 100ppm-2000ppm Alcohol
α (3000/1000) isobutane	Concentration Slope rate	≤0.6	
Standard Detecting Condition	Temp: 20℃±2℃ Humidity: 65%±5%	Vc:5V±0.1 Vh: 5V±0.1	
Preheat time	Over 24 hour		

D. Structure and configuration, basic measuring circuit

	Parts	Materials
1	Gas sensing layer	SnO_2
2	Electrode	Au
3	Electrode line	Pt
4	Heater coil	Ni-Cr alloy
5	Tubular ceramic	Al_2O_3
6	Anti-explosion network	Stainless steel gauze (SUS316 100-mesh)
7	Clamp ring	Copper plating Ni
8	Resin base	Bakelite
9	Tube Pin	Copper plating Ni

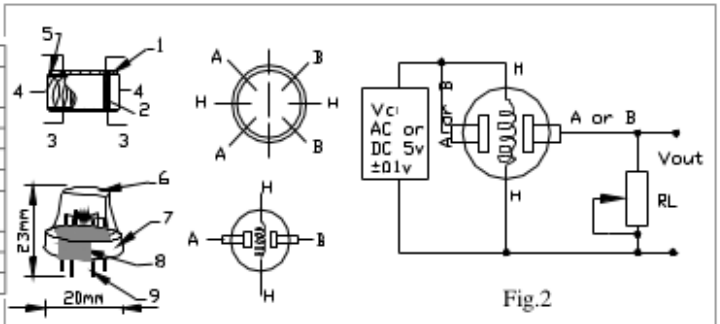
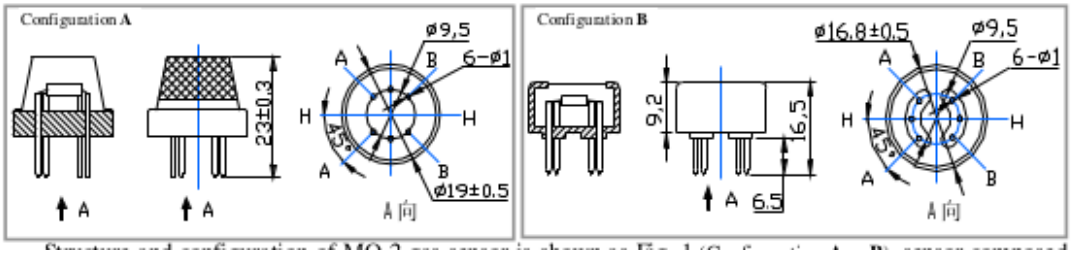


Fig. 1 Fig.2



Structure and configuration of MQ-2 gas sensor is shown as Fig. 1 (Configuration A or B), sensor composed by

PIR Sensor (#555-28027)

General Description

The PIR (Passive Infra-Red) Sensor is a pyroelectric device that detects motion by measuring changes in the infrared levels emitted by surrounding objects. This motion can be detected by checking for a high signal on a single I/O pin.

Features

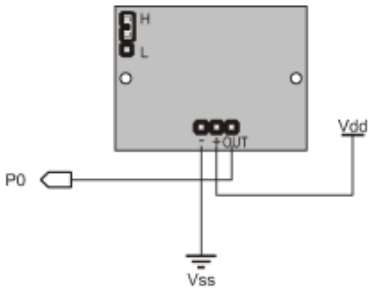
- Single bit output
- Small size makes it easy to conceal
- Compatible with all Parallax microcontrollers
- 3.3V & 5V operation with <100uA current draw

Application Ideas

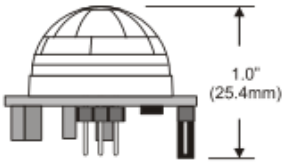
- Alarm Systems
- Halloween Props

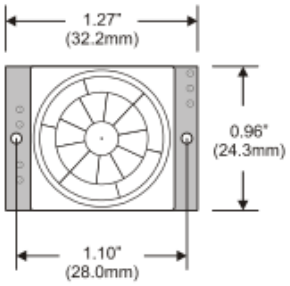
Quick Start Circuit

Note: The sensor is active high when the jumper (shown in the upper left) is in either position.



Module Dimensions





DS2Y

RATING

1. Coil data

Single side stable type

Nominal coil voltage	Pick-up voltage (at 20°C 68°F)	Drop-out voltage (at 20°C 68°F)	Nominal operating current [±10%] (at 20°C 68°F)	Coil resistance [±10%] (at 20°C 68°F)	Nominal operating power	Max. applied voltage (at 50°C 122°F)
1.5V DC	70%V or less of nominal voltage (Initial)	10%V or more of nominal voltage (Initial)	132.7mA	11.3Ω	200mW	200%V of nominal voltage
3V DC			66.7mA	45Ω		
5V DC			40mA	125Ω		
6V DC			33.3mA	180Ω		
9V DC			22.2mA	405Ω		
12V DC			16.7mA	720Ω		
24V DC			8.3mA	2,880Ω		
48V DC			6.3mA	7,680Ω	300mW	

2. Specifications

Characteristics	Item		Specifications
Contact	Arrangement		2 Form C
	Initial contact resistance, max.		Max. 50 mΩ (By voltage drop 6 V DC 1A)
	Contact material		Ag+Au clad
Rating	Max. switching power		60 W, 62.5 VA (resistive load)
	Max. switching voltage		220 V DC, 250 V AC
	Max. switching current		2 A
	Max. carrying current		3 A
	Minimum operating power		Approx. 98 mW (147 mW: 48 V)
	Nominal operating power		Approx. 200 mW (300 mW: 48 V)
Electrical characteristics	Insulation resistance (Initial)		Min. 100MΩ (at 500V DC) Measurement at same location as "Initial breakdown voltage" section.
	Breakdown voltage (Initial)	Between open contacts	750 Vrms for 1min. (Detection current: 10mA.)
		Between contact sets	1,000 Vrms for 1min. (Detection current: 10mA.)
		Between contact and coil	1,000 Vrms for 1min. (Detection current: 10mA.)
	FCC surge breakdown voltage between contacts and coil		1,500 V
	Temperature rise (at 20°C 68°F)		Max. 65°C with nominal coil voltage across coil and at nominal switching capacity
	Operate time [Set time] (at 20°C 68°F)		Approx. 4 ms [approx. 3 ms] (Nominal coil voltage applied to the coil, excluding contact bounce time.)
	Release time [Reset time] (at 20°C 68°F)		Approx. 3 ms [approx. 3 ms] (Nominal coil voltage applied to the coil, excluding contact bounce time.) (without diode)
Mechanical characteristics	Shock resistance	Functional	Min. 490 m/s² (Half-wave pulse of sine wave: 11 ms; detection time: 10μs.)
		Destructive	Min. 980 m/s² (Half-wave pulse of sine wave: 6 ms.)
	Vibration resistance	Functional	10 to 55 Hz at double amplitude of 3.3 mm (Detection time: 10μs.)
		Destructive	10 to 55 Hz at double amplitude of 5 mm
Expected life	Mechanical		Min. 10^8
	Electrical		5×10^5 (1 A 30 V DC), 10^5 (2 A 30 V DC)
Conditions	Conditions for operation, transport and storage*		Ambient temperature: –40°C to +70°C –40°F to +158°F Humidity: 5 to 85% R.H. (Not freezing and condensing at low temperature)
	Max. operating speed (at rated load)		60 cpm
Unit weight			Approx. 4g .14oz

* Refer to "6. Usage, Storage and Transport Conditions" in AMBIENT ENVIRONMENT section in Relay Technical Information.

Specifications

Chip	Broadcom BCM2835 SoC
Core architecture	ARM11
CPU	700 MHz Low Power ARM1176JZFS Applications Processor
GPU	Dual Core VideoCore IV® Multimedia Co-Processor
	Provides Open GL ES 2.0, hardware-accelerated OpenVG, and 1080p30 H.264 high-profile decode
	Capable of 1Gpixel/s, 1.5Gtexel/s or 24GFLOPs with texture filtering and DMA infrastructure
Memory	512MB SDRAM
Operating System	Boots from Micro SD card, running a version of the Linux operating system
Dimensions	85 x 56 x 17mm
Power	Micro USB socket 5V, 2A

Connectors:

Ethernet	10/100 BaseT Ethernet socket
Video Output	HDMI (rev 1.3 & 1.4)
	Composite RCA (PAL and NTSC)
Audio Output	3.5mm jack, HDMI
USB	4 x USB 2.0 Connector
GPIO Connector	40-pin 2.54 mm (100 mil) expansion header: 2x20 strip
	Providing 27 GPIO pins as well as +3.3 V, +5 V and GND supply lines
Camera Connector	15-pin MIPI Camera Serial Interface (CSI-2)
JTAG	Not populated
Display Connector	Display Serial Interface (DSI) 15 way flat flex cable connector with two data lanes and a clock lane
Memory Card Slot	SDIO

Ultrasonic Ranging Module HC - SR04

Product features:

Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of work:
(1) Using IO trigger for at least 10us high level signal,
(2) The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.
(3) IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic to returning.
Test distance = (high level time×velocity of sound (340M/S) / 2,

Wire connecting direct as following:

- 5V Supply
- Trigger Pulse Input
- Echo Pulse Output
- 0V Ground

Electric Parameter

Working Voltage	**DC 5 V**
Working Current	**15mA**
Working Frequency	**40Hz**
Max Range	**4m**
Min Range	**2cm**
MeasuringAngle	**15 degree**
Trigger Input Signal	**10uS TTL pulse**
Echo Output Signal	**Input TTL lever signal and the range in proportion**
Dimension	**45*20*15mm**