SIT210 Task 11.2P – Prototype report for the front door monitoring system:

Overview of project including:

Background

Studying from home has introduced a unique challenge for me to overcome. I have other siblings who are also studying so it is extremely important to keep things quiet or they will not be happy with me. Like most people, I like to listen to music when studying but cannot play it via speakers as my siblings may be in a class or are trying to concentrate so I have to use my noise cancelling headphones

Problem statement

Wearing these headphones leaves me unaware of my surroundings so I am unaware if there is someone at the front door as my study is located at the other end of the house. This lack of awareness within my own home can cause issues since I might not hear when someone is at the door for example the postie or a neighbour.

Requirements of the system

I require a system that is:

- Cheap
- Reliable
- Able to alert me when someone is at the door

Functional requirements:

- Be able to notify me when something is at the door
- Stream video footage to be accessed remotely
- Stream audio to be accessed remotely
- Activate the buzzer if an object has been detected for too long
- Initial activation of the buzzer will place it into an active state where any movement will trigger it again for a period of time

Non-functional requirements

- The notifications must be received within a very short timeframe
- The footage must be able to be accessed remotely
- The audio must also be able to be acceded remotely
- The system must be able to be moved to different doors around the house and still be usable

Design principles

Robustness:

I have designed my system with robustness in mind in several different ways. Firstly, there is no user input during execution of the program, meaning that users cannot send bad input to the system such as strings where numbers should be entered or nothing when something should be entered. This being said the user would still be able to directly manipulate these values before executing the script and this would cause issues but the system as is runs fine.

Flexibility:

With a few changes to the code, the specific values used such as the time before the buzzer goes off, the distance in which an object is detected or the active status time on the buzzer can be fine-tuned to work on any door. These specific values are stored in global variables and used throughout the program and in the user manual I have specified that people who know how to program can change these values otherwise they can use the default settings.

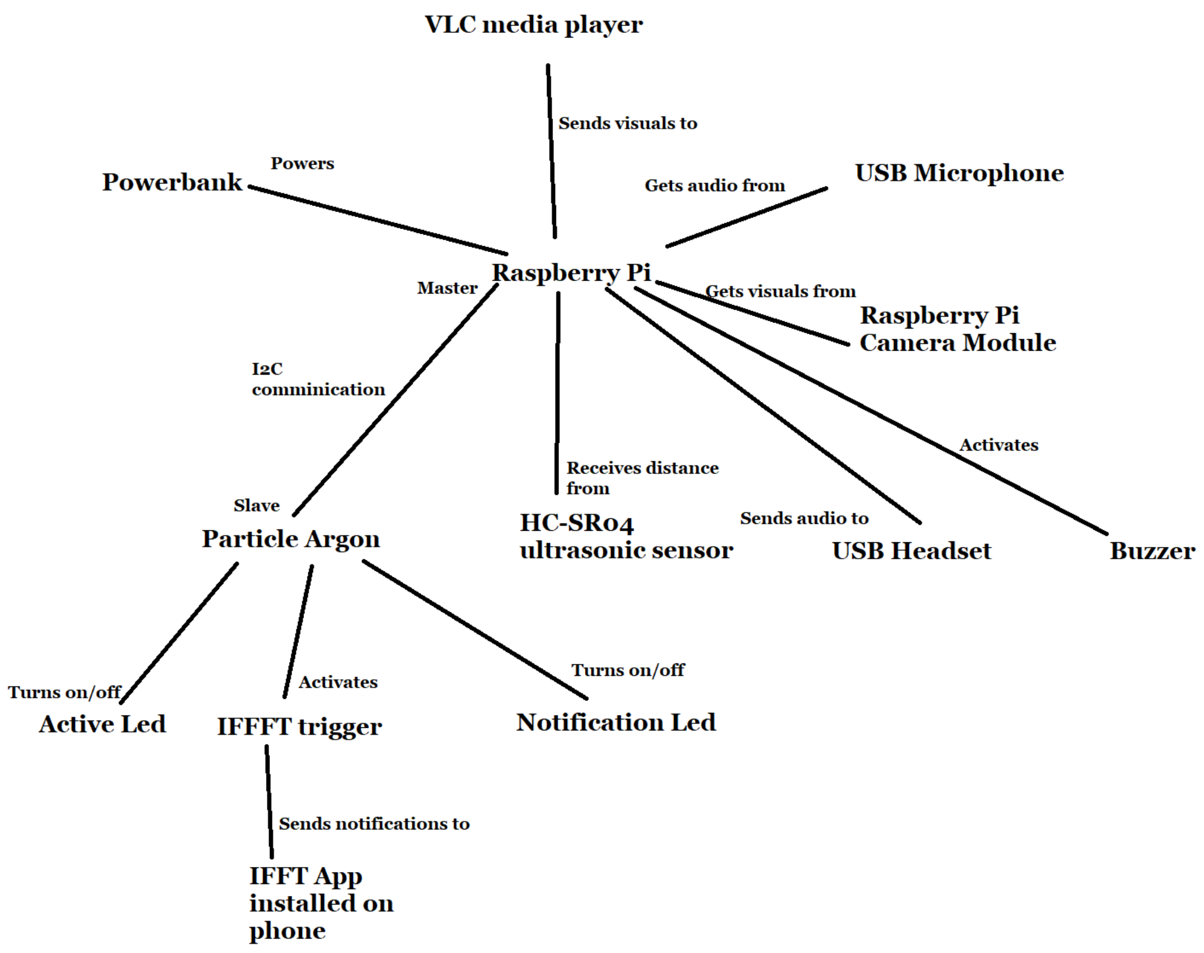
Real time response:

Since this system has applications in security it is vital to have all the components working as close to real time as possible. The system is designed to send a notification to the users phone whenever any object is detected, due to limitations of the particle cloud and the IFFFT servers it might take anywhere between 30-45 seconds from the time of detection to reach the users phone when the IFFFT app is running in the background, however, when manually checking from within the IFFFT app it provided the real time status of the system.

The video streaming is also in real time, due to the nature of any sort of live streaming there is a slight delay from the actual observed images and the actual images, the delay in this system is only about 2-3 seconds which is acceptable as the user can still gauge what is happening, it's not like they are half a minute behind the live view, in that case it would be better to go check yourself is anything is at the door.

The Audio streaming is practically real time as the audio is transmitted using the USB headphones which uses Bluetooth, meaning that there is virtually no delay between sounds entering the microphone and the sounds being transmitted through the headset.

Prototype architecture



Link to prototype code on GitHub

https://github.com/nmanoglou/SIT210-Final-Project-Code

Test approaches that I used to evaluate the system

My approaches to testing the system was very straight forward, implement a component, test for bugs, rinse and repeat. Once all the individual components were successfully implemented and were tested, I then put them all together to see how they operated, if any new bugs arose, I dealt with them.

Once I was confident that everything was working as intended began the full-scale tests. Using the user defined variables at the start of the python script, I adjusted these down to smaller values just to see how everything would work in the environment of my desk. Once I was happy that the system was working as expected I then upped these values to how they would be shipped to a customer and tested the system on my own front door.

The whole testing phase was not as straight forward as I would have hoped for, there was a major bug with the system which prevented me from using a dashboard with the system as my script needed to be kept in a while loop whereas the code for the dashboard needed to be outside the loop.

I attempted to get around this problem by moving the python code onto the argon since it had a main loop which would have ran the logic of the program nicely, allowing the dashboard to be kept by itself on the raspberry pi and any values that needed to be passed down could have been done so using the existing I2C connection. Ultimately, I ran out of time to implement this and had to resort to running the system without a dashboard to control the variables whilst the system was on.

User manual

Hardware required:

- Raspberry Pi Model 3b+
- Particle Argon with Wi-Fi antenna and supplied USB to micro USB cable
- HC-SR04 ultrasonic sensor
- Adafruit 5V buzzer
- Raspberry Pi camera module
- USB microphone
- 2x LEDs
- 2x Resistors
- USB headset
- Jumper cables (Male to Female and Male to Male)
- Solderless breadboard (mini or full size both work)

Software required:

- Raspbian operating system installed onto an SD card
- VLC media player library installed on the raspberry pi
- VLC media player installed on either your computer of phone
- Particle web IDE
- Any python IDE which works on the raspberry pi (I used Thonny)
- IFFFT app installed onto your phone
- VNC viewer for mobile or computer (preferred) to remotely access raspberry pi

Wiring tutorial:

Raspberry Pi:

- Place camera module into dedicated camera port next to 3.5mm headphone jack
- Place USB microphone and USB headset receiver into open USB ports in the Raspberry Pi
- Using the portable power bank, connect the 5V 2A output USB Port on the power bank to a open USB port on the raspberry pi and this will provide power to the whole system (make sure that it is 5V at 2A supplied or you might destroy your device)

Buzzer:

- Place the female end of one of the jumper cables on pin 8 (GPIO 14) of the raspberry pi and the male end into a pin which is the same row as the positive terminal of the buzzer
- Connect the female end of a jumper cable to a free ground pin on the raspberry pi and the male end to a pin in the same row as the negative terminal of the buzzer

Particle Argon:

- Using a USB to Micro USB cable, place the USB side into an open port on the Raspberry Pi and the micro USB side into the USB port on the particle argon, this will allow it to be powered when the system is powered
- Attach the Wi-Fi antenna that came with the system to the WIFI pin on the argon
- Place the argon on the breadboard in such as way the USB port is facing away from the breadboard and the pins are as far back as they can go into the breadboard while being in a hole, ensure that there is at least 1 column of free pins on either side of the argon.
- Connect the ground column of the breadboard to a pin which is in the same row as the ground pin on the argon using a male-to-male jumper cable

I2C connection:

- Place the female end of a jumper cable on one of the GPIO ground pins and the other male end into the ground column of the breadboard
- Place the female end of another jumper cable on pin 3 of the Raspberry Pi (GPIO 2) and the male end of the jumper cable in a pin in the same row as the SDA pin of the argon
- Place the female end of a jumper cable on pin 5 of the Raspberry Pi (GPIO 3) and the male end of the jumper cable in a pin on the same row as the SCL pin of the argon

Notification LED:

- Place the led in such a way that the 2 prongs are on different rows to each other
- The row containing the shorter end is the negative end, place one end of a resistor in the pin beside the pin containing the negative prong of the led and the other end into a spare pin in that row, just leave 1 spare pin hole free in that row
- Using that spare pin next to the resistor, use a male-to male jumper cable to connect that pin to the ground column in the breadboard
- Using a male to male jumper cable, place one end in a pin in the same row of the longer prong of the led, the other end will be placed in a pin that is adjacent to pin D6

Active LED:

- Place the led in such a way that the 2 prongs are on different rows to each other
- The row containing the shorter end is the negative end, place one end of a resistor in the pin beside the pin containing the negative prong of the led and the other end into a spare pin in that row, just leave 1 spare pin hole free in that row
- Using that spare pin next to the resistor, use a male-to male jumper cable to connect that pin to the ground column in the breadboard
- Using a male to male jumper cable, place one end in a pin in the same row of the longer prong of the led, the other end will be placed in a pin that is adjacent to pin D4

HC-SR04 ultrasonic sensor:

- Place the sensor in a way that the 2 protruding cylinder type structures of the device are facing away from the breadboard
- Place the female end of a jumper cable to a free ground pin on the Raspberry Pi and the other end in a pin in the same row as the ground pin on the sensor
- Place the female end of another jumper cable to pin 12 of the Raspberry pi (GPIO 18) and the male end of the cable into a pin that is in the same row as the echo pin of the sensor
- Using the female end of another cable, place it on pin 11 of the Raspberry pi (GPIO 17) and the male end of the cable into a pin that is in the same row as the trig pin of the sensor
- Finally place the female end of another cable into pin 1 of the raspberry pi (3.3V) and the male end of the cable into a pin in the same row as the VCC pin on the sensor

Prerequisites:

- VNC player is installed on your remote computer and is connected to raspberry pi to allow remote operation
- VLC media player is installed on your remote computer (get it at: https://www.videolan.org/vlc/index.html)
- Sd card with Raspbian operation system is up and running on the raspberry pi
- Check out the GitHub repository for this project, it will have the code as well as the streaming command you will use in the next few steps
- USB microphone and headset receiver is plugged into open USB ports on raspberry pi

Setting up the video stream:

- The first thing you need to do is enable your camera in the Raspberry pi settings, the best way would be to go to your terminal and type the command '*sudo raspi-config'* this will take you to the raspberry pi configuration page, select the *interfacing options* and enable the camera, your raspberry pi might need to restart
- Go back to your terminal and type in the following command to install VLC media player on the raspberry pi: '*sudo apt-get install vlc*'
- Once that has completed and type the following command: '*vcgencmd get_camera*' This will return 2 states, your device compatibility with camera modules, and if a camera is detected by the system. You want to ensure that both these states return a 1, indicating that they are fully operational. If the detected state returns 0 check to see if the camera is properly connected, the supported state should always return 1 since this model of the raspberry pi has the dedicated camera port.

- Once everything is working ok then you can run this final command to enable to the stream, using the last command on the camera command file on the GitHub repository, run this in your terminal and check that the output is saying something along the lines of buffering and showing a percentage, this indicates that the system is working, if you see output along the lines of 'nothing to play' then check if your camera is being detected properly.

Accessing the video stream:

- On your remote computer launch VLC media player
- Select the 'media' tab up the top then select 'open network stream', alternatively on windows press Ctrl + N
- For the address will be in the following format:
  http://{ Your Raspberry Pi IP address }:8160
- The stream should start automatically
- For more information about how the video streaming works check out the following YouTube video: https://www.youtube.com/watch?v=JjPsW-7FUng

Setting up the audio stream:

- The first thing to do is to check out the audio stream command file on the GitHub repository for a list of command to use in the next few steps
- On your raspberry pi right click on your sound bar up on the top right corner and make sure that your audio input is set to the USB microphone and the audio output is set to the USB headset
- Afterwards open up a terminal and use the 1st command listed to enable your microphone
- Following that run the second command in order to get your IP address, you will need this for the actual streaming command
- Using your IP address enter the 3rd command, substituting the 'Your RPI IP Address part' for the IP address received from using the 2nd command and enter the password of your raspberry pi device when prompted, ensure that you remove the curly braces from the command before running it.
- Turn on your USB headset and listen away to everything that the microphone picks up (Note: while not everyone has USB headsets a normal set or earphones can be used instead by plugging the AUX end into the 3.5mm headphone jack port on the raspberry pi to verify that this works, providing you have changed your sound settings in the prior part above to this device)
- For more information on how this works and to test before using the system, consider checking out this link: https://blog.mutsuda.com/raspberry-pi-into-an-audio-spying-device-7a56e7a9090e

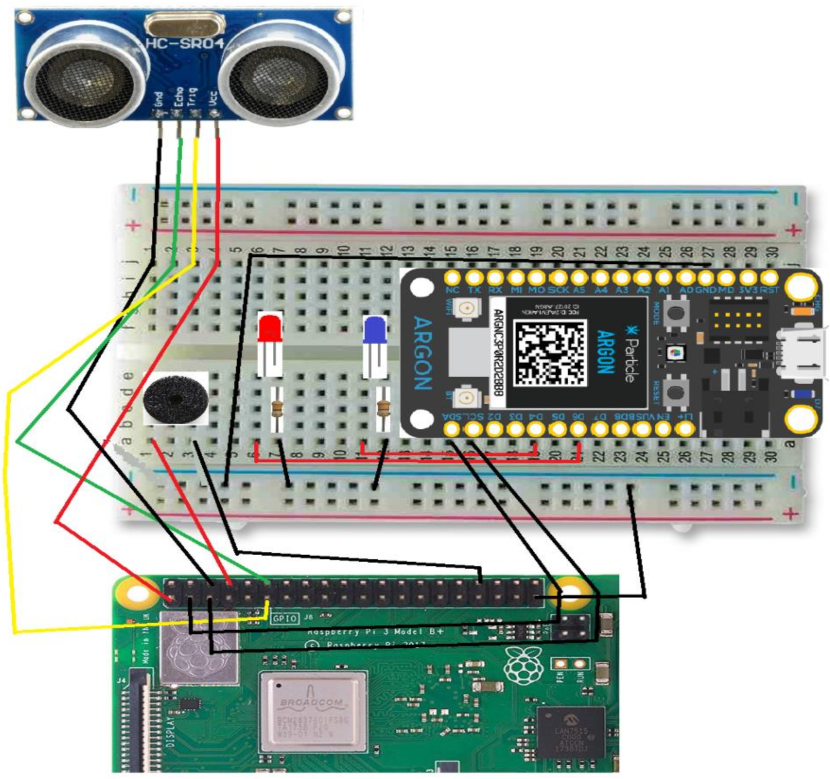
Setting up the system for usage:

- Now everything is set up and ready to go, to get the system working you will first power on the device using the power bank
- You will need to access the GitHub repository on the raspberry pi in order to download the files for the system and the streaming commands

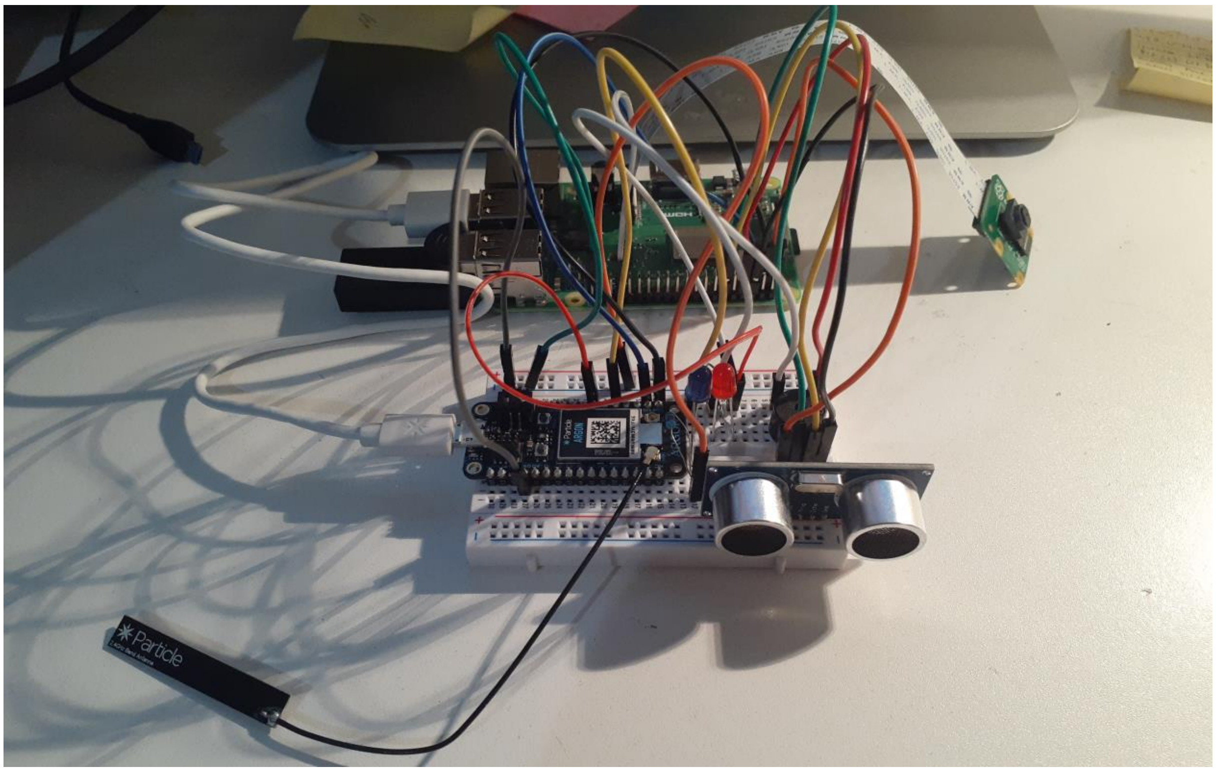
- Once you have downloaded the contents of the repository you will need to flash the *project.ino* file to the particle argon using the particle web IDE or the command line
- Next you will go and follow the steps to get the video stream set up
- Afterwards you will go and follow the steps to get the audio stream set up using a different terminal window than the one used for video streaming
- Finally, you can open the *Project.py* file which contains the code to run on the raspberry pi and press run when you open it using the Thonny IDE
- To stop the program, you can press Ctrl + C to stop it to exit the program but you will need to reopen the press run again to start it back up

What the system looks like:

Circuit diagram



Real solution:



Conclusion and recommendation for extension:

Overall, I didn’t expect to come up with what I did at the end of the day, this system was a much better idea than my original and I am glad I decided to pursue it. The biggest issue came with the dashboard, it would have been a great feature to have but I ran out of time to implement it. Another issue I had was the fact that the audio streaming is reliant on a USB headset, originally I wanted it to be streamed to my computers speakers however I was unable to get this to work so the headset was the next best option, this also made it impossible to demonstrate since I couldn’t adjust the volume of the streamed audio to be loud enough to be recorded.

If I were to do this project again firstly I would come up with my project idea much sooner than I did, if I had my idea finalised before the mid-trimester break this would have given me an extra week or 2 to order the parts and begin work on the project much sooner than the end of week 6, this extra time would have been the edge I needed to be able to work out some the dashboard and audio streaming issues which would ultimately have improved the overall quality of then system. Another thing I would do would be to use a different camera module since the once I used actually broke towards the end of development as I was recording my project teaching case so I had to use some left over footage from my project demo since I recorded that prior to the teaching case.