Bird Nest Box Monitoring Using Raspberry Pi and Python

Luke Bray - B00100787

March 6, 2019

1 Abstract

2 Acknowledgements

Contents

List of Figures

List of Tables

3 Introduction

3.1 Background

Man-made nest boxes are vital for native cavity-nesting birds. One of the main reasons for this is the increased competition for natural cavity spaces. Non-native invasive species such as Starlings and House Sparrows as well as increased urban sprawl have reduced the number of available natural cavities. The Irish landscape is one that is heavily agricultural and changes in agricultural policies and practices have provoked losses in biological diversity[4]. Therefore as the Irish landscape and policies governing it change it becomes ever more important to monitor the populations of all birds to see how they are faring[5].

Monitoring a nest box is important for many reasons. First of all it can be interesting to see what kind of species is occupying the box. However this can be achieved by simply observing the coming and going of the box inhabitants. One of the biggest threats to nest boxes can be invasive species. This can include larger bird species and small mammals as well as insects and spiders. In this case it is helpful to have a view into the box. This can be achieved by physically inspecting the nest box however this can be intrusive and disturb the wildlife inside. Thus arises a need for a remote monitoring system using a non-intrusive camera.

Maintaining a healthy habitat is important for the wildlife inside the nest box. Different species of bird are responsive to changes in environment at varying levels and some environmental factors can be critical to the survival of some species[6]. It is for this reason that some simple monitoring equipment would be useful to include in any kind of nest box monitoring system.

All of the data that will be gathered is not very useful if it cannot be stored and displayed in a meaningful way. One of the most convenient ways to view data is through a graphical means and the particular data being collected in this project may be of interest to the public and anybody who has an interest in wildlife. Therefore it will be necessary to find a suitable way to store the data and display it in a way that is accessible and engaging.

3.2 Aims and Objectives

From the background information provided above it becomes clear that there are definitely two main aims of this project. There will also be a third aim that is secondary to the function of the project however is equally important.

The first aim of this project is to have an effective and useful monitoring system. This first aim can be broken down into multiple objectives.

- To use appropriate hardware to capture the most important and fundamental data. In this context hardware refers to the actual computing device as well as the sensors that will be used to collect the data. Hardware that is used to power the device is also included in this objective as this will be important to ensure the completion of the objective.

- To have efficient and appropriate software running on the hardware. The software is important as it is what controls how the data is processed, stored and displayed. It will have to be robust enough that it can handle

large amounts of data and it should be designed in a way that allows the aim to be achieved.

- To design a system that is unobtrusive to the wildlife that will potentially inhabit or are currently inhabiting the nest box. If the device is too obtrusive then it is unlikely that birds will actually use it as a habitat and therefore no data can be collected. Similarly if the hardware and software are not well designed then regular maintenance will have to be performed which could lead to intruding on any wildlife that may be inhabiting the nest box and rendering the habitat useless.

The second aim of this project is to display the collected data in a suitable way. This second aim can be broken down into multiple objectives.

- To display the sensor data in a graph or table that is easy to read and interpret. Historical data should also be available for analysis.
- To display the video footage captured by the camera. It is important that the footage is displayed live and if possible alongside the sensor data.
- To make the data easily available to anybody who would like to view it. This will mean hosting the data online.
- To capture the data without significant loss. The software will need to account for periods of time in case an internet connection cannot be obtained and it should be an objective to capture all data without significant loss.

The third aim of this project is employ well established software design approaches. This third aim can be broken down into multiple objectives.

- To allow for accountability and risk analysis when developing
- To successfully identify the key functionalities of my software
- To abide by a software development process and have thorough design documentation before any implementation is done.
- To have a proper software testing process in place that will minimise bugs and errors at implementation
- To design a system that allows for future expansion in case I find that the project could be expanded.
- To accurately evaluate any strengths and weaknesses in the software

4 Literature Review

4.1 Introduction

Birds have been used as tools all over the world for centuries for things such as hunting, entertainment and delivering messages. In the modern world bird-watching is an already a popular activity and it's recent increase in popularity has helped to integrate research into birds, bird conservation and socio-economics development[7]. In China alone as of 2010 there were in excess of

20,000 birdwatchers[7]. The hobby is essentially the monitoring of different bird species and recording information such as the time of year, the species of the bird and how many birds there are. Many countries have organisations where birdwatchers can submit this data and this leads to an overall picture of the welfare of a countries native species.

A parallel to this rise in birdwatchers over recent years has been the emergence of the large number physical objects which are connected to the internet, commonly known as the *Internet of Things*. These devices can play a large role in our daily lives and in 2010 the number of devices connected to the internet surpassed the population of humans on Earth[8] and by 2020 it is estimated that the number of IoT smart objects deployed globally will reach 50 billion entities[9].

When we see the clear trend upwards in both birdwatching and smart devices connected to the IoT it becomes a natural step to combine the two and create a smart, connected device that can bring the hobby of birdwatching and all of the benefits it provides into the digital world. Birdwatching is essentially a hobby where monitoring takes place, albeit manually. It therefore seems natural with the help of small, powerful computers and all of the sensors available for them to build a monitoring device.

In this literature review I will explore some other monitoring projects which used technology similar to that I intend to use. Creating a bird monitoring device will present some challenges such as connectivity and power supply and I will attempt to gain a better understanding of how I might cope with these challenges in my monitoring device.

4.2 Raspberry Pi for Video Capture

The main function of the device will be to capture a live video feed in the nest box to see what species of bird are nesting inside and some of their behaviours. One of the considerations to be taken when capturing video is how it will be transferred. Traditional monitoring camera systems such as Closed Circuit Television (CCTV) have poor video quality and rate of transfer. It is important for the device to be used for this paper to have high definition video with low latency times. One article[10] set out to verify that high quality video could be transferred with low delay times using small, low-powered computers such as the Raspberry Pi.

The aims of the study were to create a system that meets the following requirements:

- A low delay from source to sink of <200ms to provide an as close to live as possible viewing experience
- High definition video content of at least 1280x720 at 25 frames per second
- Runs on cheap, resource-constrained device such as Raspberry Pi

To achieve a stable transmission of video that does not saturate a network, video compression is usually required. One of the most popular video compression formats that is used is the H.264 and that is what is used in the article discussed here. Fortunately the Raspberry Pi camera modules (both IR and non-IR) were designed with high quality video streaming in mind and as such

capture video as raw H.264 video stream[11]. The H.264 codec was created to provide good video quality at lower bit rates which makes it an ideal format for internet video streaming[12]. The article discussed found that from source to sink, the average delay was 181ms at the specified resolution which met the requirements of their study.

One problem with the above study is that all of their tests were done using wired connection since the Raspberry Pi 2B that they used didn't have the ability to transmit via wireless connection. Given the remote nature of the monitoring system for this project the connection will most likely have to be wireless and I would expect that delay time will be greater than 181ms. However the study did find that 90% of the total delay is due to the encoding and decoding of the video so I believe the increase in delay using a wireless transmission will be minimal. I am also happy that very low delay times are not a key requirement of this project.

4.3 Raspberry Pi as Monitoring Device

While live video feed is intended to be the main function of this device, the secondary function is to monitor the environment the device is placed in and measurements will be recorded by the device to achieve this.

Many other papers have attempted to do this. One such paper, which was intended to create an IoT urban based climate monitoring system, found that due to its low cost and high reliability the Raspberry Pi was a good choice for a small, low powered monitoring system[13]. The study used a Raspberry Pi 2 Model B along with 5 sensors to provide a broad overview of the surrounding environment. The system used Adafruit IO which provides different libraries that are used for implementing an IoT device. The libraries are available in many different languages however the one that was used here was Python. Adafruit IO can also generate a dashboard automatically containing graphs of the outputs from the various sensors which was useful to the study to display data in a meaningful, easy-to-understand way.

One of the issues that the article discussed above did not address is how the device is powered. This is an important aspect of a monitoring device as they need to be constantly running to collect meaningful data.

Another study[14] used an Arduino board to house the sensing units which then transferred the data via Zigbee to the Raspberry Pi 3 which uploaded and processed the data onto the internet. The advantage of this is that the Arduino can be used as a separate sensing unit and the Pi can be used as the computing unit. This will increase performance across the two devices although it does add complications because the data has to be transferred to the Pi and then also to the internet. This wouldn't be a problem for a device that doesn't make use of Pi Camera module because the Pi can be safely located indoors however for this project the Raspberry Pi must be located close to the nest box so that the Pi Camera can be attached inside.

Another problem with the above study is that it does not mention how the Arduino sensor module or Raspberry Pi are powered. They have stated that the system could potentially run for months without human intervention but have not discussed how the devices could be powered for this amount of time. This is something that I will try to discover in the next section.

4.4 Powering the Raspberry Pi

Raspberry Pi is powered by a +5.1V micro USB port and it is recommended to purchase a 2.5A power supply for stability reasons. According to the documentation a typical Raspberry Pi 3 Model B can use between 700-1000mA[11].

One paper set out to achieve an accurate estimation of the Raspberry Pi's power consumption and see if they could reduce power consumption by optimizing the software running on the platform[15]. In the study they set the Pi up to run only essential operations and specifically wrote lightweight utilization software to reduce power consumption while still achieving their goal. The monitor was not even permitted to write to the SD card, instead writing the results to RAM and copying the data after the monitoring had finished. The above was done with the intention of being able to monitor system utilisation without the process of monitoring affecting results.

The study then used software called *cpulimit* to generate load on the system. This software creates an infinite loop which adds numbers until the CPU load is at the desired limit. The study ended up with 900,000 power measurements which were combined and plotted logarithmically in a heat map fashion on the graph below[15]:

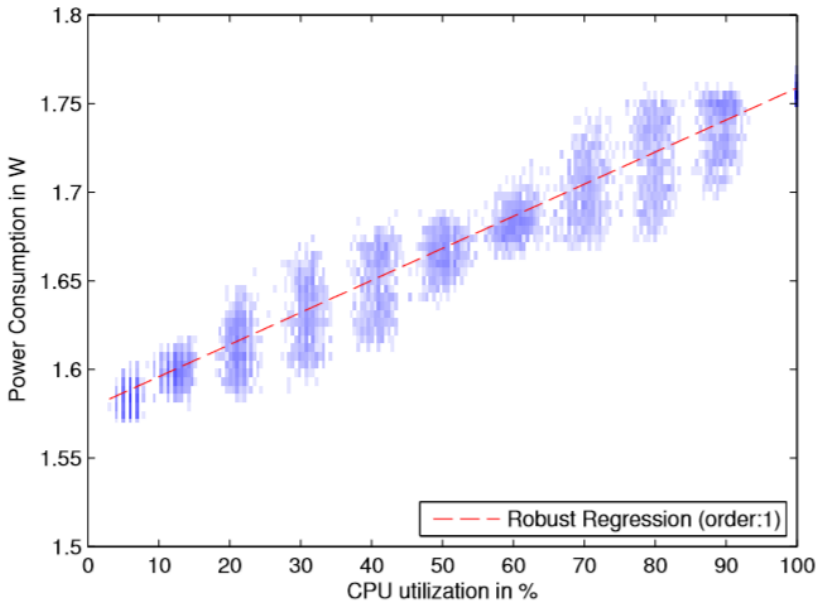

Figure 1: Power consumption vs. CPU utilisation

The graph shows that CPU usage and power consumption have a linear relationship and as CPU usage reaches 100% power consumption is at approximately 1.75W or 1750mA. This far exceeds what is stated in the documentation from Raspberry Pi although it can be argued that a Raspberry Pi will very rarely be close to 100% CPU usage for any extended period of time. The study also attempted to model power consumption during network transmissions and found that using Ethernet consumed far less power. At 50MBit/s Ethernet had

upload/download power consumption of 0.3W and 0.32W respectively. While WiFi used over 1.5W on both upload and download.

The study above is a good example of how to measure the power consumption of a small device and could be easily adapted to measure devices other than the Raspberry Pi. It does highlight the need to think carefully about the types of transfer methods used as these can clearly have a significant impact on power consumption.

4.4.1 Solar Power

It seems that the most efficient way to power a monitoring system would be to use a renewable energy source. The most accessible and affordable is solar power. A solar power system has three main components:

1. *The Solar Panel* - This is what most people think of when they think of solar power. It is a panel that uses the photovoltaic effect to convert sunlight and to electricity and is the core of any solar power system.

2. *The Solar Controller* - This is an important part of any solar power system. It's role is to regulate the amount of charge coming from the panel to the battery and prevent overcharging[16]. Devices can also be charged directly from the solar controller with some versions even including 2.5A USB ports which are ideal for charging devices such as Raspberry Pi.

3. *The Battery* - This part of the solar power system is optional. If you have a device which is being charged directly then there will be no need for this part of the system however it could be important in climates where there is not a lot of sunlight or if you have a device that needs to be powered overnight.

With the three components above a typical solar power system circuit diagram might look like the one below[17]:

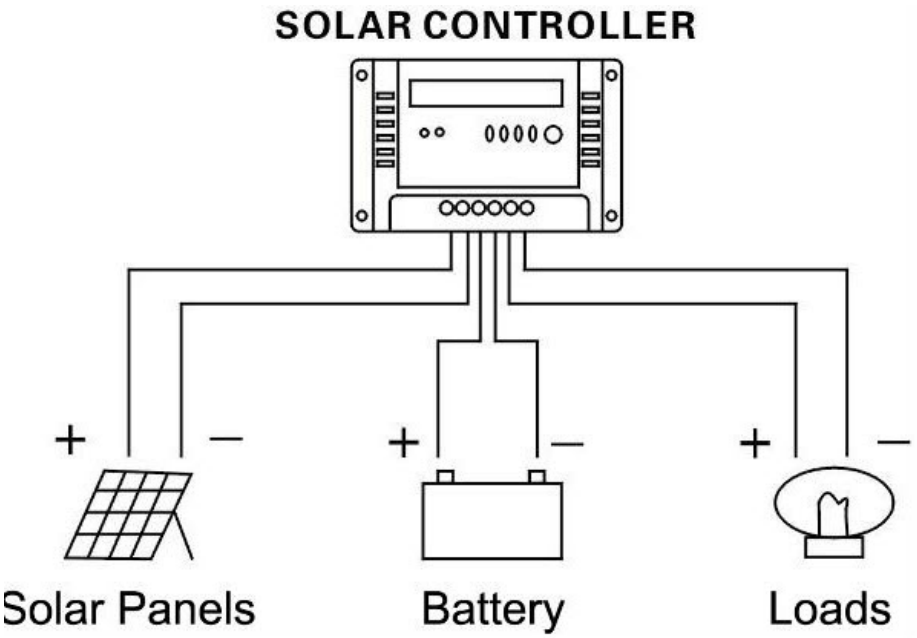

Figure 2: A basic solar circuit

4.5 Raspberry Pi Connectivity

There are multiple options available for connectivity to the Raspberry Pi and 3 of the main methods will be explored below.

4.5.1 WiFi

The latest model of the Raspberry Pi comes with a built in 2.4GHz 802.11n wireless adaptor, allowing it to transmit data without any need to external adaptors or wires. There are many WiFi standards available however the most prominent is 802.11n and it is found in most wireless devices such as smartphones, tablets and laptops. There is a competitor called 802.11ac which was developed to handle larger files and faster speeds. Several studies, including one by Ruckuswireless[18], have found that 802.11ac is a superior connectivity standard to 802.11n however the low cost of 802.11n has meant it has become the favoured standard and this is why it appears in the Raspberry Pi.

802.11n can transfer data with throughput up to 300Mbps and over distances of 70 metres in open structure environment and 250 metres in open air environment[19]. This makes it ideal for transfer of relatively small amounts of data from a monitoring station however power consumption and reliability of WiFi in general still remain a cause for concern. WiFi also has quite weak security and the size of a WiFi network must remain relatively small with a maximum of around 16 devices on any normal network.

4.5.2 Wired Transfer

Raspberry Pi 3 Model B comes with 1 Gigabit Ethernet over USB2.0 port. The use of USB2.0 means that the throughput is limited to 300MBps - the same as WiFi. We have already seen in[15] that Ethernet connections use significantly less power than WiFi however because the transfer is via USB2.0 we do not see the speed advantages that Ethernet connections generally offer. We will still get a faster connection because there will be no loss as there is with WiFi and using a wired connection means we get more stable speeds. Wired Ethernet connections are also more reliable that WiFi connections due to the removal of any chance of network interference from other transmissions.

4.5.3 Zigbee Wireless Protocol

Zigbee is wireless specification based on IEEE 802.15.4 that was first seen in 2007 and is widely used in wireless monitoring devices due to it's low-cost and low-power requirements. This protocol describes only the MAC and physical layers[20]. Zigbee uses small, low-powered and low-cost radios to transmit signal. The data transfer rate of Zigbee is 250Kbits/sec. It can operate on 2 regional frequency bands (868MHz in Europe, 915MHz in America) and 1 global 2.4GHz frequency band[21].

A strength of Zigbee is that it can transmit signal at distances of up to 100m between sender and receiver. One criticism has been that their needs to be line of site between sender and receiver however there have been papers that suggest signal strength may not be affected by obstacles. Idoudi, 2013 used the Received Signal Strength Indicator (RSSI) to show that there was little other than distance between two nodes that negatively affected RSSI values and that distance had a more negative effect on RSSI than any obstacles in the way[22]. However the study did not state the distance between two nodes when the obstacles were placed between them. RSSI indicates the strength of the signal at the receiving node.

Interference with Zigbee from WiFi can also occur and this is one of the main drawbacks of Zigbee, along with the rate of transfer. Zigbee uses 16 channels within the 2.4GHz band and these directly overlap with the 3 channels used by WiFi. This is shown in the figures below [23]:

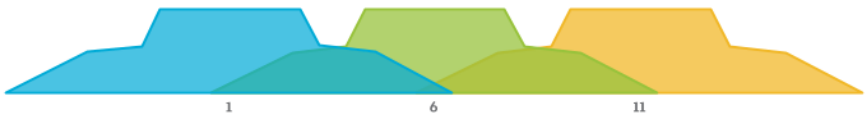

Figure 3: 2.4GHz WiFi Channels

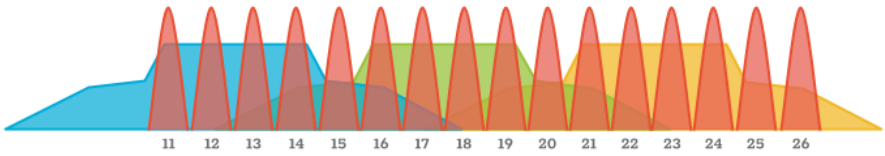

Figure 4: 2.4GHz Zigbee Channels

Usually the Zigbee network is the network that will lose packets. The side-band lobes of the WiFi signal can easily drown out the less powerful Zigbee signals. This is one of the biggest problems with Zigbee and careful planning is required when it will be used in close proximity with any WiFi-enabled devices.

4.6 Python Web Frameworks

A key aspect of this project is to display the collected data in a way that is appealing and accessible. This will most likely mean displaying the data on a web page. However due to the rapid evolution of the World Wide Web it has become difficult to utilise all of the technologies required for good web development[24]. Using a web framework can be a good way to tie the various technologies together to allow for quicker development and also to facilitate future expansion more easily. This section will look at three Python web frameworks and discuss some of their features.

4.6.1 Django[1]

Django is a free, open-source Python web framework. It is a Model View Controller (MVC) style framework and is focused on never having to repeat yourself when programming. Django is a framework that allows for quick development and minimal repetition of code. It also ensures that your app is secure against a multitude of vulnerabilities. This richness of features is also one of the main criticisms against Django. While the vast array of features may be ideal for a large scale project, they are not needed for a smaller scale project. Django is also considered to be quite monolithic and not hugely flexible. You get features such as admin panels and database interfaces out of the box which means it is not nearly as customisable as some other frameworks.

4.6.2 Flask[2]

Flask is seen by many to be the main rival to Django. Like Django, it is a MVC style framework however it is not nearly as feature-rich and provides only the more essential features. This more minimalist design means that Flask may be better suited to smaller project however there are some large projects such as Netflix and Pinterest which are built on Flask. This may be an indicator that there is also good scalability with Flask.

4.6.3 Dash[3]

Dash is a micro-framework that is built on top of Flask and Plotly.js. The framework includes various UI elements and allows you to tie them together

with Python code while also retaining all of the benefits of using a MVC framework like Flask. Dash is specially built to be used for building analytical web applications and this makes it ideal for a project where data is to be presented and analysed.

5 Methodology

5.1 Introduction

Stating importance of having a good methodology and how not having a clear methodology can be bad

5.2 Data Collection

How data will be collected. Will discuss what will be done with the data and why the data collected is important

5.2.1 Video Data

How video data will be used and why it is the most important data to collect. Discuss some steps to make sure the data is of good quality

5.2.2 Environmental Data

How will this data be presented to the end user and how will it be interpreted? How can I make sure that this data is meaningful and is of good quality?

5.3 Development Method

This will contain info about the SDLC method I will use and a brief explanation of what this method involves

5.3.1 Planning

Importance of planning. Some methods that can be used to plan effectively and efficiently

5.3.2 Designing

Importance of good design and some ways to ensure a good design

5.3.3 Building

Which specifications will be followed when building the physical housing for the product. Why is this important and how do I make sure that I stick to these specifications. What technical drawings are needed

5.3.4 Testing

Importance of testing and how testing will be carried out. What kind of tests need to be run to ensure a complete product

5.3.5 Developing

Which language will be used. Some best practices for software development and the importance of using international and industry-recognised coding practices and standards

5.3.6 Implementing

How to implement a finished product. Is this done incrementally and step by step or is the final version of the product released once it is finished? How is a product updated and maintained and the importance of this from a developers point of view and a consumers

5.4 SDLC Models

5.4.1 The Waterfall Model

What is the Waterfall Model and some of it's advantages and disadvantages

5.4.2 Agile Development

What is Agile development and what are it's advantages and disadvantages. How do they relate to my project?

5.5 Conclusion

What can I conclude from the above? My thoughts on how all the above information pertains to my project and some things I will have to be especially concious of during this project

6 System Requirements and Design

6.1 Introduction

Outline the key requirements of the system and some of the main design problems to overcome

6.2 Build Schematics

This section will show technical drawings and electrical diagrams

6.2.1 Nest Box

How the nest box will be constructed and the specifications of the nest box. Technical blueprints will be included in this section showing the layout and physical construction

6.2.2 Device

6.3 Hardware Components

What components will be used

6.3.1 Raspberry Pi

Specifications about the Pi model to be used in this project. Some of the challenges and limitations of the model to be used and also why this particular model was selected for the project

6.3.2 Sensors

Information and specifications of the sensors and camera to be used for this project. Why they were selected and how they will be connected. What output will they provide. This section will contain electrical and wiring diagrams

6.4 Software

6.4.1 Operating Systems

The OS that will be used and how it is installed. Why was it selected and what are it's main features. How will these be leveraged to produce a successful product

6.4.2 Data Processing

How the data will be processed and what software needs to be written to do this. This section will contain some UML diagrams to demonstrate this

6.4.3 Data Visualisation

Will discuss how the collected data will be displayed to the end user and the software that will do this. Will explain how this is done using UML diagrams

7 Testing and Evaluation

7.1 Introduction

Why testing is important for this product and what tests need to be done to ensure a good prototype

7.2 Testing Methodology

Howe to be scientific about testing and create complete and useful tests. The testing timeline will be outlined here with one successful test meaning another can now be run. This schedule will be defined in this section

7.3 Device Functionality

Functions and features of the device. How the device is intended to perform

7.4 Evaluation

How the device performed on each of the tests that were conducted. What areas now need to be improved to move towards a more complete prototype

7.5 Implementation of the Prototype

How the prototype will be implemented. What considerations need to be taken when implementing and what conclusions can we draw from the testing that affect the implementation of the prototype

8 Further Work

How might the device be improved? What extra features could be included next time and how might a different SDLC have an effect on the design and functionality of this product

9 Conclusions

What did I learn while doing this project. What went wrong and what was done well. Is the device created as useful as was initially thought? Is the way I did this project the most effective way to achieve the intended result?

References

[1] Django Software Foundation. Django, March 2019.

[2] Armin Ronacher. Flask, March 2019.

[3] Plotly. Dash, March 2019.

[4] Pauline Pierret and Frédéric Jiguet. The potential virtue of garden bird feeders: More birds in citizen backyards close to intensive agricultural landscapes. *Biological Conservation*, 222:14 – 20, 2018.

[5] Birdwatch Ireland. Garden bird survey, November 2018.

[6] Janice Wormworth and Dr. Karl Mallon. Bird species and climate change. Technical report, Climate Risk Pty Ltd, 2006.

[7] Zhijun Ma, Yixin Cheng, Junyan Wang, and Xinghua Fu. The rapid development of birdwatching in mainland china: a new force for bird study and conservation. *Bird Conservation International*, 23(2):259 – 269, 2013.

[8] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys Tutorials*, 17(4):2347–2376, Fourthquarter 2015.

[9] Perumal Arumuga, Krishnamoorthy Baskaran, and Suleman Khalid Rai. Implementation of effective and low-cost building monitoring system(bms) using raspberry pi. *Energy Procedia*, 143:179–185, 12 2017.

[10] Ulf Jennehag, Stefan Forsstrom, and Federico Fiordigigli. Low delay video streaming on the internet of things using raspberry pi. *Electronics*, 5(4):60, Sep 2016.

[11] Raspberry Pi Foundation. *raspivid documentation*. Raspberry Pi Foundation, 3 2017. Documentation available https://github.com/raspberrypi/documentation.

[12] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the h.264/avc video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):560–576, July 2003.

[13] R. Shete and S. Agrawal. Iot based urban climate monitoring using raspberry pi. pages 2008–2012, April 2016.

[14] Sriyanka and S. R. Patil. Smart environmental monitoring through internet of things (iot) using raspberrypi 3. pages 595–600, Sept 2017.

[15] F. Kaup, P. Gottschling, and D. Hausheer. Powerpi: Measuring and modeling the power consumption of the raspberry pi. pages 236–243, Sept 2014.

[16] Energy Matters. What is a solar regulator/charge controller, February 2019.

[17] EdgeFX. Solar charge controller working using microcontroller, February 2019.

[18] Ruckus Wireless. 802.11ac: Next steps to next-generation wifi. Technical Report 2, Ruckus Wireless, Ruckus Wireless Inc., 350 West Java Drive, Sunnyvale, CA 94089 USA, 10 2014. An optional note.

[19] T. Kaewkiriya. Performance comparison of wi-fi ieee 802.11ac and wi-fi ieee 802.11n. In *2017 2nd International Conference on Communication Systems, Computing and IT Applications (CSCITA)*, pages 235–240, April 2017.

[20] T. Elarabi, V. Deep, and C. K. Rai. Design and simulation of state-of-art zigbee transmitter for iot wireless devices. pages 297–300, Dec 2015.

[21] Zigbee 3.0 specification. `https://www.zigbee.org/zigbee-for-developers/zigbee-3-0/`. Accessed: 2018-12-10.

[22] M. Idoudi, H. Elkhorchani, and K. Grayaa. Performance evaluation of wireless sensor networks based on zigbee technology in smart home. pages 1–4, March 2013.

[23] Zigbee and wifi coexistence. `https://www.metageek.com/training/resources/zigbee-wifi-coexistence.html`. Accessed: 2018-12-10.

[24] Dragos-Paul Pop and Adam Altar. Designing an mvc model for rapid web application development. *Procedia Engineering*, 69:1172–1179, 2014.