# Literature Review

This section will introduce the wireless technologies used, the Python programming language, its libraries used, as well as the protocols and frameworks used to send, store, and host the data recorded.

## Python

Due to the speed at which programs can develop, its availability across almost all platforms, the range of libraries, and how convenient modules are to import/export, Python is the obvious choice for this project.

Reasons to choose Python:

* High-level – allows for 20 lines of code to be written in five
* Flexible – can be run on almost any platform
* Modules – scripts can be written and easily included in another
* Libraries – there are quite a few

[1]

Many different libraries will be used to fulfil the needs of this project and will be installed using the Python package installer (pip), which installs the libraries easily, handles different library versions, and is available on Mac, Windows and Linux. An example from the Pip website is shown below:

#### pip install SomePackage # latest version

#### pip install SomePackage==1.0.4 # specific version

#### pip install 'SomePackage>=1.0.4' # minimum version

When developing on Python, much of the problems are to do with incompatible library dependencies, the shell commands above install the default package followed by a specific version and minimum version option too.

### Dispatcher

This Python library is used as a centralised service for delivering messages to objects.  It enables the configuration of functions to enable the receipt of signals from senders; this is very important for this application as updates made across the Z-wave network must be handled, the dispatcher can use any Python object as the sender, saving the need for typecasting.

### NumPy

This library gives the programmer use of powerful multi-dimensional array objects, mathematics tools, and many more capabilities that act as a substitute for the more familiar MATLAB. It also supplies much of the dependencies that other libraries in this report require, including PeakUtils, which is passed a NumPy ndarray and uses its differentiation functions to obtain peaks. [<https://www.numpy.org/>]

### Matplotlib

Further likening Python to MATLAB, this 2D plotting library produces highly customisable, graphs which can are very useful for the gathering of data. Multiple data series may be plotted alongside each other, making comparisons between each is far more convenient when testing raw data. [docs]

### PeakUtils

This library presents several functions to detect peaks on 1D data. Using numpy functions, the indexes, baseline and Gaussian fitting curve can be applied to an array, the resolution can also be increased. The indexes function gives great flexibility, giving the programmer control over the amplitude threshold, the minimum time between peaks, and options for threshold interpretation. [docs]

### Pandas

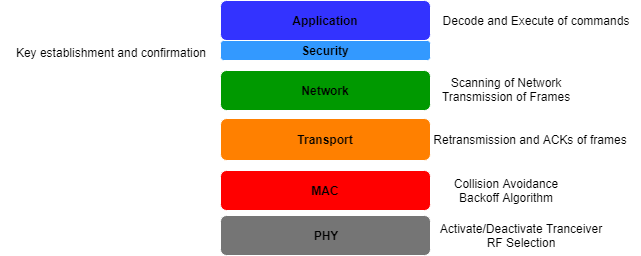
Pandas (**P**ython **Da**ta **An**alysis) is an open-source library available on GitHub and aims to deliver high-performance, accessible data structures, and data analysis tools. It offers two main data structures: DataFrame (df) and Series (ps). The df is a two-dimensional array, like a spreadsheet, this is very useful for database querying and is included in the InfluxDB pip library for this reason. It offers powerful data cleaning tools when exporting resource into a script, specifying missing values and column name, saving time and increasing program efficiency.

[<https://www.tutorialspoint.com/python_pandas/python_pandas_introduction_to_data_structures.htm>]

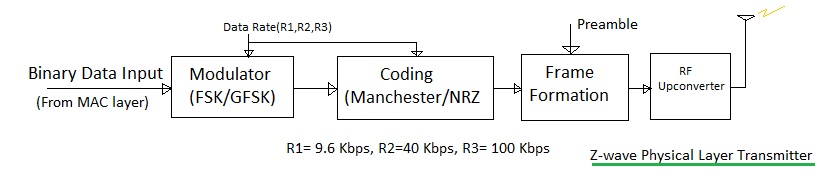
## Z-Wave

Z-wave is used as a home automation technology and offers far more benefits than traditional Wi-Fi networks. Operating at ISM frequencies, this prevents interference from other IoT technologies like Wi-Fi, that typically operate at ~2.4GHz. It also operates on a mesh network offering a maximum of 232 nodes (slaves) on a network, which would rule it out for larger projects, but multiple master devices may operate with a network bridge. Its new 700 series has a maximum range of 100m and operates at such low-power it claims coin cell devices last for up to ten years. Its interoperability is also highly desirable in an ever-changing system where new sensors could be added every week, in addition to its backward compatibility, preventing issues with out-of-date devices. Z-Wave's automatic address assignment adds these new devices without additional configuration; after installing a system for a user no additional maintenance is required in this respect, much of the smart plugs which are used to monitor appliance energy usage, are Z-wave compatible too. [13]

### Protocol Stack

A Z-wave network is divided into domains, the number of domains is limited by the 32-bit Home ID and each domain has its own Home ID. These domains have a set of nodes all connected to a controller (master), a maximum of 232 is possible and are identified by their 8-bit Node-ID which is unique to that domain; nodes also may intercommunicate on the same domain to extend the range of the devices. It uses four levels of the OSI model as shown in Figure njsvsdkj.

1. The application layer handles the receipt and implementation of commands, its frame format consists of the header, which indicates the MAC address of the source/destination, the command class, the command itself, and its parameters.
2. The network layer handles the frame routing node-to-node as well as between controllers. There are two types of packets available:
   * Routed Singlecast – which permits internode hopping until it reaches its destination
   * Routed Acknowledgment – simply informs the controller it has reached its destination
3. The transport layer’s responsibility is to acknowledge segments, retransmit lost ones, wake up devices, and to authenticate segment sources.
4. The MAC layer includes a collision avoidance algorithm to prevent nodes from transmitting simultaneously. This is achieved by notifying other nodes in the domain that they are transmitting; those nodes then delay their transmission to accommodate.
5. Onto the physical layer, Z-wave operates at 868.3MHz (EU) with data rates being 9.6-100kbps.  Z-Wave uses Gaussian frequency-shift keying (GFSK) modulation rather than the typical BPSK, this affects the recoverability with higher SNR. The Gaussian approach supports the higher 100kbps bit rate while still using the same 27ppm crystal; GFSK achieves this by passing the baseband pulse through a filter which smooths it and limits the modulated spectral width or put simply less of the unwanted frequencies are excluded. The block diagram in Figure nlkvknl, shows the modulation and encoding to Manchester, for 9.6kbps and NRZ for the two others.

[ <http://www.rfwireless-world.com/Tutorials/z-wave-physical-layer.html>]

To obtain symbol synchronisation, this encoded signal must have a preamble bit sequence added, now the receiver may scan all available channels. This is then followed by a start of frame field, the data itself, and the end of the frame.

[13]

[G.9959]

The frame format for some of the mentioned layers is shown in Figure ksdv.

GetÂ readyÂ forÂ keyÂ establishment
Ready
NonceÂ request
NonceÂ valueÂ 
EncryptedÂ networkÂ keyÂ â Kn
NonceÂ request
NonceÂ valueÂ 
Enc...[hack Zwave]

### OpenZWave

This open-source API is a reverse engineered Z-wave protocol, hiding much of the complexities involved with it but still supporting much of the features needed. It is available on GitHub and offers multiple supported languages and a Python wrapper. It facilitates the connection of a Z-wave enabled USB controller (UZB) with available nodes in a domain, it also supplies examples on the Git repository to get started and is available on multiple platforms including the Raspberry Pi and Windows.

Z-wave is also compatible with home automation platforms like OpenHAB and HomeAssistant, allowing a user to control and monitor devices easily, from an array of devices through the available binding. But to gain full access of the Z-wave protocol, OZW must be used, as Z-wave is operated by the Z-wave alliance making it partially proprietary. The Python library can be installed using pip and can be used like any other installed library.

## InfluxDB

Influx is an open-source, time-series database (TSDB) designed for high throughput inserting and real-time querying. A TSDB is one that samples data with respect to time, this could be anything from data rate from a server to the current read from a sensor, TSDBs has the primary key set to a timestamp by default which is measured in nanoseconds, giving a high level of precision to sampling. Influx makes use of Log-Structured Merge Trees (LSM) as data structures enabling high speed inserting and querying while still providing indexing. TSDB also offers downsampling, allowing mentioned high-precision data to be summarised after a certain period, preventing old data occupying storage unnecessarily, saving the need for the programmer to design data deletion techniques. InfluxDB is part of the TICK stack (Telegraf, Influx, Chronograf, Kapacitor) developed by Influx Data, which offer data visualisation, monitoring, and alerting to optimise a system’s management. As with most databases, Influx supports secure access, users can be added in the Influx CLI console.

[<https://www.influxdata.com/time-series-platform/>]

Influx uses a HTTP API to communicate with a database, using standard HTTP response codes as well as authentication offered by the protocol, all responses are returned in JSON format, increasing efficiency and interoperability with other frameworks. A typical HTTP request is shown below, where the “query” is written in InfluxQL:

#### GET <http://localhost:8086/query>

#### 

The GET is telling HTTP to send something, the URL points to the local IP and the default port number for InfluxDB.

[<https://docs.influxdata.com/influxdb/v1.7/tools/api/#write-http-endpoint>]

With the huge increase in IoT devices, the need for high precision, the self-regulating database is vital to the effectiveness of a system. In a home environment, there could be in excess of 50 sensors recording real-time data, Influx is capable of inserting millions of points to a database per second; these sensors are typically connected to a low-power microcontroller (MCU) with limited storage, the downsampling offered could save the system on extra storage costs.

Influx is available on multiple platforms and languages making it ideal for an IoT solution where the device could range from cloud servers to coin cell sensors. This system is Python as the development language and the Raspberry Pi as the gateway device, both are supported on Influx, with the library available on GitHub and can easily installed using pip.

## Grafana

To visualise and analyse the stored data, Grafana is used, which is an open source graphing tool available online for displaying time-series data. It supports multiple databases, including Influx, and offers database query like control over the plotted series, these series can be graphed in a multitude of ways from bar charts to scatters graphs with options for time intervals to manipulate the granularity of the data. Alerts may be added at certain threshold values, with the option of an automatic email in that event as well as supporting many different types of authentication, including Google, GitHub and many more.

Grafana is multi-platform, available on Linux and Windows, the installation recommends users to be added for security and the changing of the default password. After installation, databases used can be added as data sources in the online tool, which is accessible on the port 3000. All configuration is available via dashboards which can contain multiple graphs from different sources entirely, this enables the user to monitor completely different measurements from one dashboard or compile similar ones into specific ones.

[IEEExplore] [Grafana]

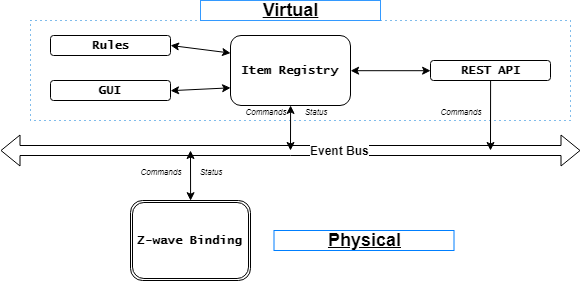
## OpenHAB

Like previous frameworks, OpenHAB2 (OH2) is open-source and multiplatform, supporting an array of technologies, it offers a central point of communication in a home network to monitor sensors and control devices, it runs on most hardware so does not need the cloud but can be integrated. Integrates home automation devices into the smart home, such as electric motors controlling a shed door with a coin-cell sensor on one graphical user interface available on both browsers and mobile applications. Bindings may be added that can be thought of as add-ons which enable the inclusion of other services such as IFTTT (IF This Then That), triggering external services to be invoked when certain condition(s) are met, both condition and trigger are user-defined-increasing flexibility. These bindings spread into the mainstream too with Amazon Alexa and Google Home working together with OH2 and for the programmer, it supports a REST API too.

OH2 aims towards a home server environment with a dedicated RPi operating system called OpenHABian (OpenHAB + Raspbian) available, which configures the system and gives a user-friendly interface. It can also be installed on default Raspbian, which gives further control over what packages can be installed on the RPi and how they may interact. OpenHAB runs as a server and listens for HTTP requests on port 8080, to prevent the mimicking of a server an SSL certificate is saved to that machine but unfortunately, OH2 does not officially support authentication, so multiple users cannot access the OH2 instance remotely.

[<https://www.openhab.org/docs/installation/security.html>]

There is a certain naming convention to this framework that must be understood before making use of the framework, hardware devices that are added to an OH2 server are called “Things”, these are not exclusively physical devices and can be some other resource like a web service, these can be set up to the user’s preference. Channels are the implementation of this configuration, a Thing may have multiple channels, just like a device may have multiple functions. “Items” are objects that reside in the virtual layer and send commands, these are sent through the Channel to the Thing, diagram in Figure jsdjsahj. Bridges are Things added to access other Things of that type, these include controllers for a device and web service authentication setup. Rules are triggered by many different events, but one is a command from the event bus in the form of an Item, which manipulates the state of a Thing, this can be very useful for applying one event too many Things or many several conditions for one action.

A user accesses a channel linked to an item available on the online dashboard, this is sent to the bus where the virtual “item” is mapped to a physical “thing” in the Z-wave binding, the command is executed, and the thing’s status is returned.

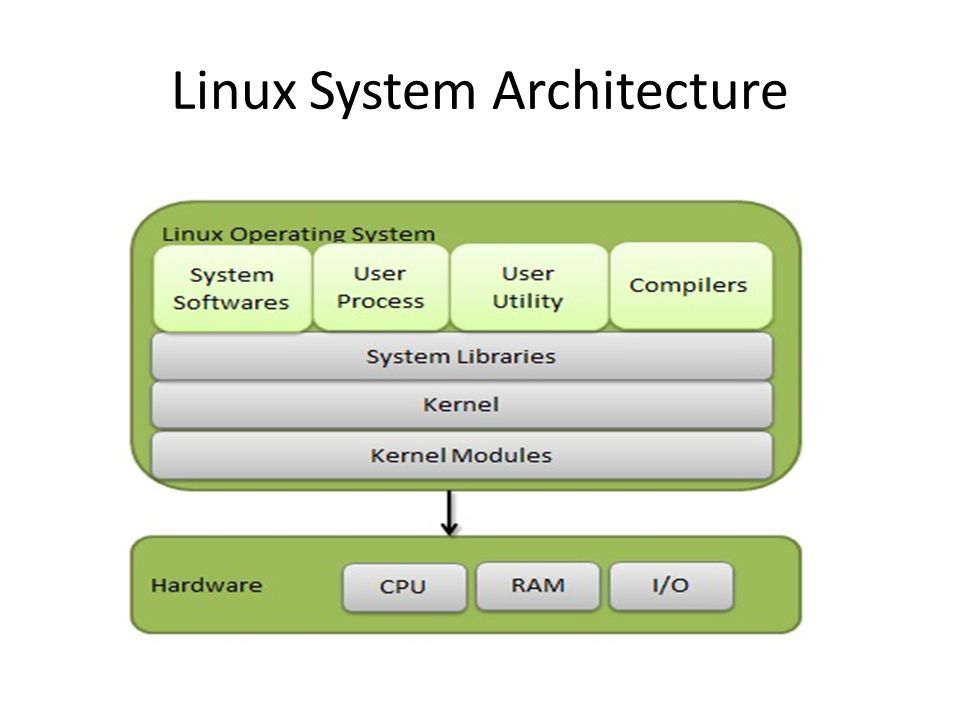
[<https://www.openhab.org/docs/concepts/things.html>]

OH2 is developed in Java and uses Apache Karaf and Eclipse Equinox for runtime with Jetty for the HTTP server. Both Equinox and Karaf run on the Open Services Gateway Initiative (OSGi) architecture which is used for developing modular applications in Java using the container concept. Systems scale faster in distributed applications with this approach and could be adopted for a future HEMS of this kind.

[<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8506917>]

## Linux

The most used operating system in the world with the greatest flexibility of any other, Linux’s distributions are available in many flavours from smart TVs to supercomputers, it offers the feature-rich GUIs on Android devices or command line server machines and is all open source. Open source allows the freedom to run any program as well to analyse, edit, and publish its source code. Developers have availed of this freedom to make multiple distributions including Chrome OS, Debian (Raspbian), and much of the software running on modern data centres; its resource efficiency makes it the obvious choice for many service providers like Facebook and Google. Its architecture is shown in Figure sger.

Just like any other operating system, Linux is based off a monolithic kernel, which supports interfacing between the hardware and software, task scheduling, memory management and many more low-level functionalities required in a computer. Linux is known for its security, stability an

d efficiency making it very desirable for embedded devices like ARM and supports virtualisation too, making it a viable option for a server solution. Device driver support is another merit of Linux with tiny MCUs like the one in a Raspberry Pi as well as the standard x64 ones available on desktops and computers alike.

[<https://developer.ibm.com/articles/l-linux-kernel/>]

[<http://slideplayer.com/slide/9660838/31/images/2/Linux+System+Architecture.jpg>]

It has garnered many IoT developers with its support for multiple databases, and languages like Python and InfluxDB which require great flexibility to benefit fully from their capabilities. Its powerful Terminal is well known for its wide range of uses which include running user scripts and daemons, installing packages and navigating directories trumping Windows with its intuitive and comprehensive command selection. Its file structure cordons all user data into the /home directory while all others are related to OS and application resources like binaries and libraries; this simplifies the development of applications with all devices and processes are available as files, further empowering the programmer. An example of Terminal use is shown below with the user first followed by the current directory, the command, the parameter and its options.

#### user@laptop:~/dir$ command file.type –options

All commands used in Linux have documentation available online as well as with the use of the man command (manual), which takes the command as parameter returning useful directions for use.

## Gaussian Filter

## Modbus

## Hardware

### Raspberry Pi

The Raspberry Pi 3 (RP3) is a minicomputer about the size of a credit card, with the processing capabilities of a small computer but with the flexibility of a microcontroller. The RP3 supports HDMI, USB, Wi-Fi, Secure Shell (SSH), Ethernet and many more technologies as well as having 40 GPIO pins, giving it access to web services and databases and in turn giving them the low-level operations needed to for IoT applications. RP3 is commonly used by consumers as web servers, network attached storage (NAS) and media centres, this flexibility is enabled by the Raspbian OS that can be installed, which can also host development tools like Python and Linux’s Terminal. RP3 offers more capabilities than its predecessor, with the Cortex-A53’s quad-core chip improving multitasking and reducing power consumption to around 7 watts it makes it ideal for running a home system which could be taking hundreds of recordings a second. Its size makes it very suitable for a domestic installation where it can be installed anywhere where a micro-USB cable can fit, and its low noise provides a non-disruptive option.

Raspbian supports almost all packages available on desktop Linux distributions; the powerful command line available broadens the systems networking capabilities supporting multiple interfaces, needed to connect Modbus in addition to SSH, which allows remote access to a device, lessening the need for a bulky GUI slowing the system. Raspbian can be installed with mentioned GUI or without (Lite version) which only displays a CLI making for leaner system. Accessing the Pi remotely is also favourable for speed of development, the code can be written and tested on a faster machine like a desktop version of Linux such as Ubuntu and then deployed, leaving only the potential compatibility issues between the two distributions.

### Smart Plug

This is a Z-wave enabled device which is inserted into a 230V wall socket for an appliance to plug into it. It can monitor the power used and control when current can flow. It is a user-centric device that has a small footprint and minimal aesthetic features, making it a favourable option for domestic use, it is Z-wave plus enabled, having a further range with less power usage. The plug can send cumulative kWh and instantaneous:

* Current
* Watts
* Voltage

In addition to the above, the node ID, interval time, and appropriate units for each value are sent; these are all accessed through the OZW with the read register function or from the OH2 dashboard, both can also turn this alter this device’s state. This report requires some sort of failsafe in the event of a power failure, this device retains all configuration after powering off; upon power up, it will still abide to the rules set out by the system.

The user manual for the smart plug is inadequate for this project, especially in how it communicates with the Z-wave controller and the resolution of the power meter. After emailing the manufacturer, Aeotec, for the specification sheet and details regarding the power meter precision. Below are the power specifications:

Minimum sensor reading:2W

Power reading percentage of error:

* P < 300W: ±3W
* P ≥ 300W: ±1%

The sheet sent also gave an example of how to interrogate a Z-wave device, all done through the command class of OZW as well as a detailed Excel spreadsheet.

[Excel sheet from Aeotec]

### Z-Stick

This acts as the physical gateway between a Z-wave node and software like OH2 and OZW, it is the central point in a Z-wave network, like a Wi-Fi router connecting local devices to the public internet. As mentioned earlier in the Z-wave subsection, it connects up to 232 nodes with the option to add an additional bridge; it is contained in a USB dongle, often called a UZB and is inserted into the HEMS, it can have nodes added with or without insertion, with its 100mAh battery and comes with an action button for pairing devices. Like the Smart plug, it too has Z-wave plus extending device battery life and communicating up to 150 metres. It can be accessed through the serial ports on the host device with some additional configuration.

# References

[1] <https://www.pythonforbeginners.com/learn-python/what-is-python/>

[13] M. Yassein, W. Mardani and A. Khalil, "Smart homes automation using Z-wave protocol - IEEE Conference Publication", *Ieeexplore.ieee.org*, 2016. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7745306. [Accessed: 04- Feb- 2019].