University of Strathclyde

Low cost dynamic vision sensor for fall detection system

19520 – Group I

Andrew Burr
Lewis Brown
Graeme Fitzpatrick
Daniel Nadejde
Kirsty Purden

16 December 2019

Abstract

This interim report covers the progress made on developing a fall detection system using a low-cost DVS sensor at the end of semester one. The system aims to detect if a person has fallen, communicate with them, and help them receive the appropriate assistance without the need for a physical sensor or intrusive monitoring. The design and operation of the system was agreed upon, and the work was split up between the five group members. Background research was then carried out, and progress on each individual section was made. The goals for end of the semester were met, and the system is expected to meet the specifications outlined at the beginning of the project, and to be completed on time.

Table of Contents

1 Introduction

Falls are one of the most significant cause of injury and death in the population, particularly those over the age of 65 [1]. This is due to health problems associated with aging such as poor vision and balance that increase the likelihood of an older person falling [2]. In addition, it is imperative that the person receives timely help, as the risk of death increases if they are left on the floor for an extended period of time [3]. While many solutions to this problem exist, they need the user to be wearing a device at all times, and often require a monthly subscription. This project aims to create a non-invasive method of detecting falls and assessing the user's condition using low-cost DVS.

This system will detect movement in a room, capture the movement using a low-cost DVS sensor, and determine if the user has fallen. If a fall has been detected, then the system will communicate with the user, and either a named contact or emergency services will be alerted. A DVS sensor was chosen, as it is considered less intrusive than a conventional camera, and is less computationally intensive. It is also important that the user is able to communicate with the user intuitively, as they may be distressed due to the fall.

The aims of this project are to design, build, and test a system that is capable of detecting falls using a low-cost DVS sensor. This will be done by researching the relevant theory and developing parts of the system in parallel. Once all components of the project have been developed and tested individually, they will be combined to form the complete fall detection system. Extensive testing will then be performed to ensure that the design works in a variety of situations.

The project should be completed by the end of semester 2, and should stay within a budget of £500. In addition to creating the fall detection system, the project should also be demonstrated at a trade show, in which users should be able to interact with the product.

This report covers the progress made from the start of the project up until the end of semester one.

2 Background

2.1 Risk of Falls

According to the World Health Organisation, falls are the second most prevalent cause of unintentional injury death worldwide, with the majority of those deaths being people older than 65 [1]. Elderly people are most at risk as they often have balance problems, vision problems, and other health conditions [2]. In addition, the injuries sustained from falls are often more severe if the individual is elderly due to health complications associated with ageing: osteoporosis increases the chances of breaking a bone and muscle weakness makes it more difficult for an older person to get up after a fall on their own [2]. The negative consequences of falls extend beyond physical injuries, with individuals often experiencing a loss of confidence and a decline in mental health [4]. Studies have also suggested that living alone increases the risk of falling [5], as receiving help after a fall is more difficult.

If a person has fallen, they may require urgent medical assistance such as an ambulance. Emergency responders can then assess their condition, and take them to hospital if required. This level of assistance is not always required however: on some occasions, a fall victim may only require a relative or friend to help them up.

2.1.1 Current Available Solutions

There several products currently on the market that help a person if they have fallen. They include bracelets and necklaces that activate either if the device senses that the person has fallen, or if a button is pressed [6]. Once the device has been activated, the individual will receive a call from a dedicated call centre, who will then act accordingly. While these systems are effective, they do have some disadvantages: they always require the device to be worn for the system to work; the devices can be large and unattractive; and they require the user to pay a subscription to the call centre. An ideal solution would not require a physical sensor or a subscription.

Efforts have been made to use smartphones to detect falls [7], but the accuracy of the sensors on the phones limits their use. Furthermore, since elderly people are generally reluctant to adopt new technology [8], a solution that relies on smartphones is difficult to implement on a large scale.

2.2 Camera

The use of a camera to solve the problem of fall detection offers a different approach than the current available solutions. Where most current solutions are active approaches with physical devices a camera provides a passive solution, where a mounted camera can be used to monitor a room or hallway. This solves many of the issues with current solutions, there is no physical sensor that an elderly person may find unattractive. As well as this, the technology would only need to be adopted once, after installation the camera system can be forgotten about by the user.

2.2.1 Conventional Camera

A standard conventional camera is one which is frame based, meaning that it captures still images over and over very quickly, each image being one frame. This would be a limited approach to the solution as it would pose a significant security risk to store footage of someone in their home for any length of time. It would be a straightforward job to hack into such a system and copy the footage. On top of this, many potential users would find such a system unappealing from a privacy standpoint, everyone wants to feel secure and alone within their home, recording footage would go against this goal. A secondary limitation would arise from a technical viewpoint – there is a large redundancy in transmitting full frames of data across a network when the only relevant information is whether a person has fallen or not, any background beside the person is information which does not need to be transmitted.

2.2.2 Dynamic Vision Sensor

A Dynamic Vision Sensor (DVS) is a camera which only records changes in pixel values, outputting a stream of on/off values in real-time, known as “events” [9]. Due to this, the DVS is known as an event-based camera, i.e. it is not frame-based like a conventional camera. In such a system, each pixel responds asynchronously, comparing the current light intensity level to a reference and generating an event if the threshold is met [10]. This means events can trigger at any time, therefore generating small bursts of data instead of an entire frame at once. At the output of this system an image is constructed from the currently triggering pixel events, black where pixels are ‘off’ and bright where an event has been recorded. An example of such an image is shown in Figure 1.

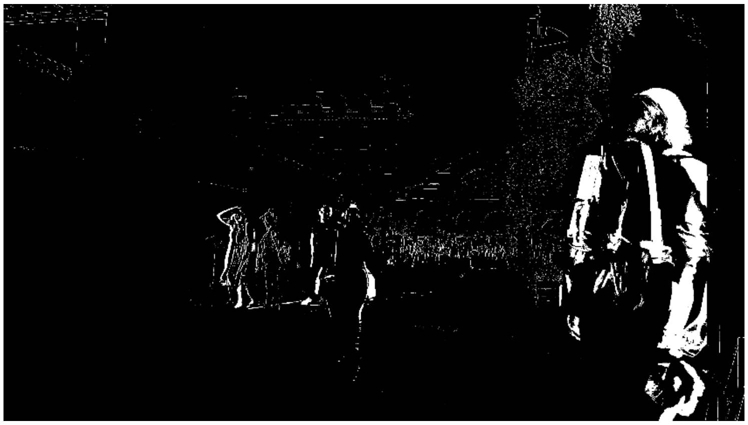


Figure 1: Example DVS Footage

2.2.2.1 Advantages over a Conventional Camera

Event-based cameras have several benefits over conventional cameras, namely in the required processing power, almost zero latency and limited stored data. There is a vast saving on processing power due to the event-based nature of the cameras - the only data transmitted is the change in pixel intensity which greatly reduces the redundant data processed from a conventional camera. This has the secondary benefit of reducing the power consumption of the system as less work is being performed. The asynchronous nature of the events allows for almost zero motion blur as the temporal resolution can be reduced to microseconds before the same pixel can next send data, allowing for very fast motion to be captured.

The former point, limited stored data, is of key importance to this project. Due to the desired implementation within a user's home, privacy must be considered. Where a conventional camera may put off a potential customer as they are reluctant to store video footage of their home, the footage captured by a DVS camera is limited to outlines and therefore considered to be more private than a regular camera as it becomes "hard to identify a person or place with only the captured images" [11].

2.2.2.2 Issues with DVS

The main issue with DVS is that it is a relatively new technology and as such is both extremely expensive and difficult to find a seller, with a market price quoted at $2700 [12]. This limits the ability to include DVS technology in a commercial product currently, with the forerunner in this market being the Samsung SmartThing Vision [11], a product which is currently only available from Australia. This device uses DVS to "capture the outline of people in its range", marketing this as a private security camera which can detect between humans and pets. DVS camera technology is clearly the future of passive sensing within the household, however the price and availability is still the most pressing issue.

2.2.3 Emulated DVS

Although true DVS systems are still difficult to acquire, this can be worked around by emulating their event-based nature, creating a "low-cost DVS" system. This approach captures data from a conventional frame-based camera and convert the differences in pixel brightness between a current frame and a reference frame into a stream of event "spikes". By comparing these spikes against a threshold an output event can be transmitted alongside the pixel location [13].

2.3 Field Programmable Gate Array

Field Programmable Gate Arrays (FPGAs) are semiconductor integrated circuits (ICs) made from silicon. In simple terms, an FPGA offers a pool of digital components (Logic gates, Look-Up Tables, Flip-Flops, etc.) which can be connected by wires. This pool of components is structured around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects (wires).

The functionality of the FPGA can be configured via hardware programming. This is achieved by writing code known as Hardware Description Language (HDL) and this is what creates the physical connections between the components. Another unique feature of the FPGAs is the reprogrammability. Their functionality can be changed as often as needed, either to improve an existing HDL design or to implement an entirely new design. This can be done "in the field" by the FPGA users, by using the development tools provided by the FPGA vendor. A different approach was taken in modern FPGAs by including application-specific dedicated hardware components such as DSP48 slices, Block RAM, Mixed-RF transceivers and soft-decision FECs for Software Defined Radio and many others. [16]

2.3.1 System on Chip

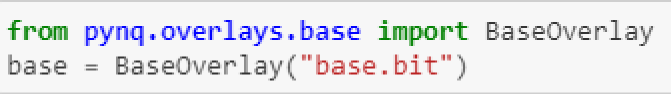
The System on Chip (SoC) device was introduced in 2011 with the Zynq-7000 series of FPGAs [17]. In such a device the traditional FPGA programmable logic (PL) is combined with a dedicated processing system (PS) [18]. All Xilinx SoCs use ARM processors, developed by the semiconductor company Arm Holdings. This PS encompasses the processor itself within the Application Processing Unit (APU), AXI interconnects to communicate between PL and PS, memory interfaces and external peripheral interfaces such as USB, Ethernet, IIC [19]. Another important feature of the SoCs is the embedded Linux OS that runs on the ARM PS.

An SoC device has numerous advantages over a traditional FPGA. SoCs can be used to enable key analytics and hardware-accelerated software algorithms. The integrated environment composed of CPU, DSP, ASSP and mixed-signal functionality on a single device, offers a high level of physical security. Another advantage of the SoCs is that it offers the best price per watt as it can achieve power and performance levels which exceed that of discrete processor and FPGA systems. Moreover, extensive libraries of industry-standard tools and IPs are available from Xilinx and so the developers can focus more on achieving their goals. Also, SoCs are more versatile as it enables customization for almost any design requirement. This versatility comes from the possibility of offloading CPU intensive tasks on the programmable logic or the capability of creating integrated peripherals from the programmable logic. [20] [21]

2.3.2 PYNQ-Z2 and PYNQ Framework

The PYNQ-Z2 is an example of a Xilinx SoC FPGA, separated from a standard SoC device through PYNQ, a Xilinx developed project to allow programming of the Zynq-7000 PS using Python [22]. This is performed through the browser-based Python environment Jupyter Notebooks. This is a powerful tool which allows code to be executed in the PYNQ-Z2's Linux image whilst also utilizing the FPGA's PL and input/output (IO) capabilities.

The PYNQ design flow relies on the use of overlays, a combined PL and PS package which is used to program the SoC, in the same way that a bitstream can be used to program a traditional FPGA. This overlay can be imported to a Jupyter notebook file as shown in Figure 2.



```
from pynq.overlays.base import BaseOverlay
base = BaseOverlay("base.bit")
```

Figure 2: Importing Base Overlay to Jupyter Notebooks [23]

The overlay shown in Figure 2 is the base overlay which comes pre-installed on the PYNQ-Z2 board. This overlay contains PL to allow for the majority of the board's IO options to be used [24]. By using this overlay, peripherals can be accessed and controlled through Python code by importing the required sections, making this a useful starting point for many designs as the PL design has been created already. A high-level overview of the base overlay architecture can be seen in Figure 3.

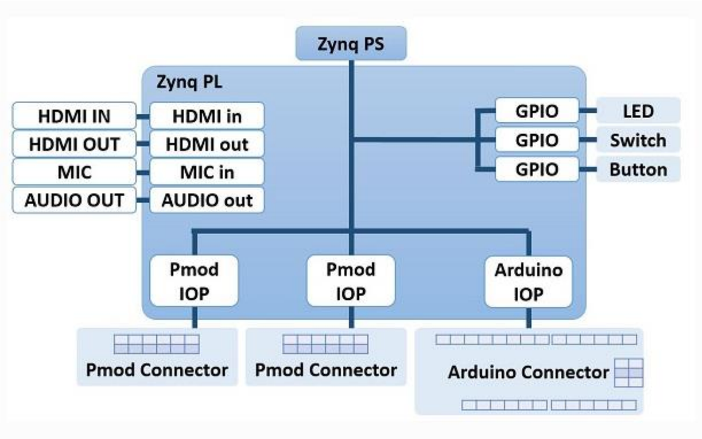


Figure 3: Base Overlay PL Design [24]

The base overlay is limited however, it is wasteful of resources when a peripheral is not useful for a design, such as audio or the Arduino. It is also often useful to incorporate new IP to a design for a specific application. Fortunately, overlays can be easily customized or designed from scratch. The design of a custom overlay is a 3-stage process.

First, a hardware block design is created for a generic application, such as image processing, using Vivado IP Integrator (IPI) design tools. Afterwards, the block diagram is validated and used to generate a bitstream. An FPGA bitstream (.bit) is a file that contains the information needed to program an FPGA board, i.e. set-up the physical connections between the components. Normally a bitstream file is all that is required to program an FPGA, but since the PYNQ board is a SoC, which uses the overlay system to configure the PL, an additional file is needed. This file is called a "Tcl" script and it is generated from the same block design that was used to create the .bit file. The script contains information that is used by the PYNQ framework to identify the Zynq system configuration and information about IP blocks like: versions, interrupts, resets, and other control signals. Having created the bitstream file and Tcl file it is now possible to move to the next stage in the overlay creation procedure.

In the second stage, the two files are copied on the SD Card of the PYNQ board and then used to program the PL via the overlay system as shown in Figure 2. After the custom overlay is successfully loaded onto the board, the next important part left to do is to test it. This is achieved by sending instructions to the Zynq PS through Jupyter Notebooks.

The final stage is about the creation of Python drivers for the IP blocks found in the overlay. These drivers basically mask the read/write requests sent to the Zynq PS via Jupyter Notebooks, with more abstract Python user-defined functions. This in turn makes the Python code easier to read and more user-friendly.

2.3.3 Vivado and IP Integrator Design Tools

The PL design for the PYNQ overlay, and for SoC FPGAs in general, is created using the IP Integrator tool, within Xilinx's design suite Vivado. Vivado is an integrated design environment (IDE) with a high degree of design automation through design, implementation, synthesis and bitstream generation [25]. IP Integrator is one of the main design tools within Vivado, creating block diagrams using IP blocks to connect hardware fabric logic to the PS system [26].

2.4 Wireless Systems

2.4.1 Software Defined Radio

Software Defined Radios (SDRs) are systems where hardware components used in traditional radios (mixers, filters, amplifiers, modulators/demodulators, etc.) are implemented through software on embedded systems. Figure 4 below shows a high-level overview of the SDR.

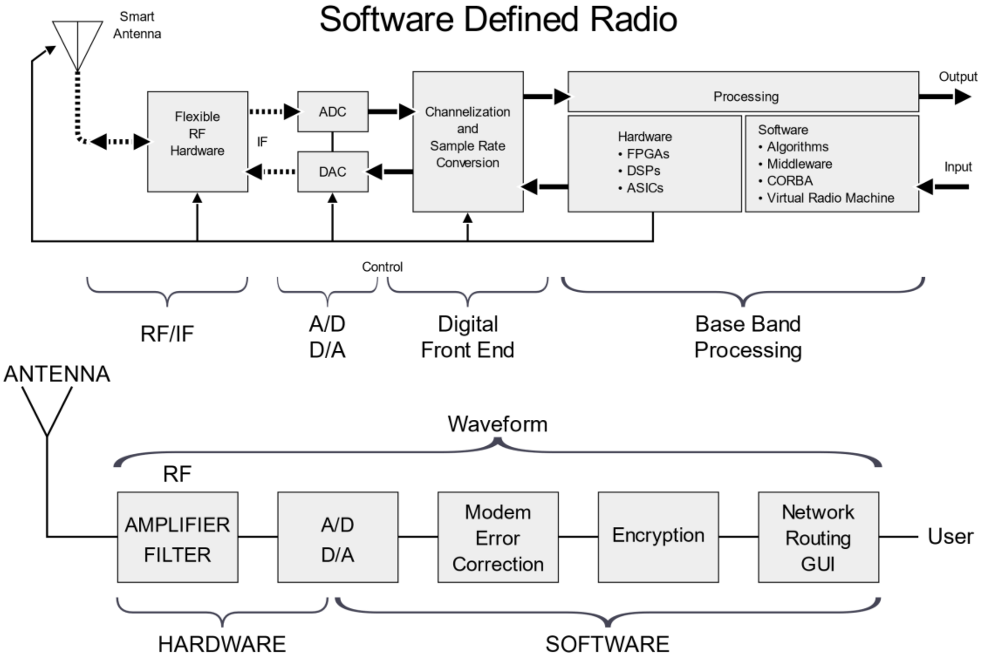


Figure 4 Software Defined Radio concept [27]

High-speed SDR receivers have a sampling frequency of 5 GHz which means that it can digitize all signals from baseband to 2.5 GHz, according to Nyquist's sampling theorem. Some of these signals are: Wi-Fi (2.4 GHz), LTE (800 MHz), GPS (1.5 GHz), Bluetooth (2.4 GHz), and FM radio (100 MHz) [28]. SDRs can be also used to transmit data over an arbitrary legal radio frequency, since some frequency bands are reserved for military use or other private applications. These radios can be implemented on FPGAs, but they are very expensive in terms of the resources required. High-speed dedicated Analog-to-Digital (ADC) and Digital-to-Analog (DAC) converters are also needed.

2.4.2 Wi-Fi

Wi-Fi refers to a wireless network that uses radio waves to provide access to the internet and network connections. It transmits at frequencies of 2.4 GHz or 5 GHz, higher than other systems that use radio frequencies to transmit information. Wi-Fi can be integrated and used in many systems and devices as it is a standardized technology which follows the 802.11 networking standards developed by the IEEE. The process of transmitting and receiving data is based on the same concept seen in Figure 4. Data is translated into a radio signal by a PC's Wi-Fi adapter and sent via an antenna to a wireless router. The router will then decode the radio signal and transmit the information to the internet through a physical Ethernet connection. [29]

2.5 Standard Digital Camera Module Interfaces

As the PYNQ-Z2 does not provide a standard conventional camera interface on the board, one must be built on the FPGA Fabric order to capture data from a camera module. To do this requires an in-depth understanding of the interface. By knowing the control interface, signal timing and the data layout of each of the pins of the interface the data can be read in and processed effectively. This requires a large amount of work for complex interfaces, so a high-speed but simple interface that can be understood and implemented quickly is preferred.

In addition, while there are many different interfacing options available, each manufacturer produces their modules with the same few interfaces. For example, the main flexible and cheap image sensors available on the market use "MIPI" Standardised interfaces [30]. This streamlines the choices for modules to be used in this project mainly to these interfaces.

2.5.1 MIPI (Mobile Industry Processor Interface)

'MIPI Alliance' is a global organization that aims to standardize many of the mobile industry interfacing standards. This means that many large-scale companies such as Samsung, Texas Instruments, ARM, Intel, Nokia, and more industry leaders, use these interfaces in their products [30]. With this, image modules are produced that use these interfaces and many smaller companies and products follow suit. MIPI interfaces have specifications for many things, but their camera interfaces the main point of interest because these make up most of the digital image module market [31].

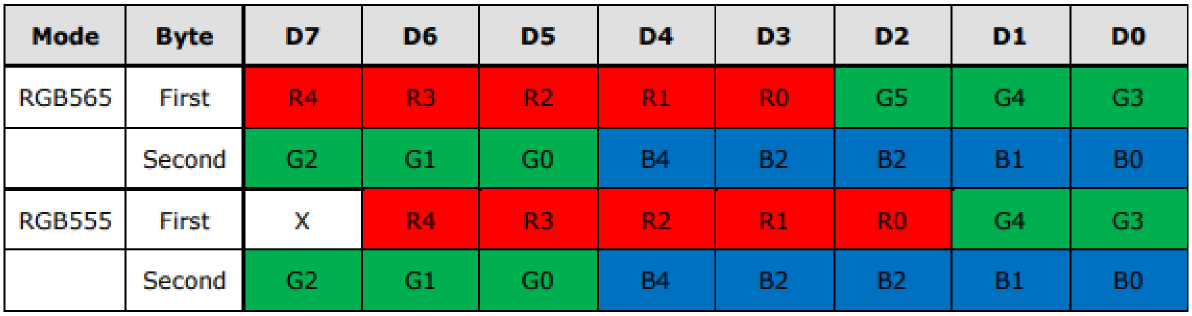
2.5.2 Camera Parallel Interface (CPI)

The Camera Parallel Interface is simple to implement manually. It comprises of an I^{2}C control bus, and parallel data signals. The I^{2}C control bus is a 2-line channel (Clock and Data) that allows a "master" processor to control the device configuration. This also allows it to be connected as one of many I^{2}C "slave" devices on a product, so it does not need its own dedicated lines for configuration.

The parallel data signals include a Vertical Sync signal (VSYNC), Horizontal Reference Signal (HREF) and Pixel clock (PCLK), along with the data lines for the pixel data. Each of these work in tandem to transfer pixel data for each frame the image sensor captures. VSYNC Deals with timing individual frames while HREF Helps time and differentiate each line of the image. PCLK Handles the timing of the pixel data being sent. For each implementation of the interface the data format may be different.

Pixel data may be sent in various formats, and with different bit depths for each channel of the pixel information.

Examples of common data formats include RGB565 and RGB555, where Red Green and Blue pixel information is sent over a 2-byte packet in a 16- or 15-bit mode. As a byte is sent every Pixel Clock Cycle, 2 bytes are needed to send over the pixel data, however, the parallel data lines mean that each byte is received all at once, rather than bit at a time, theoretically increasing data transfer speed. An example of these data formats is given below in Figure 5



Mode	Byte	D7	D6	D5	D4	D3	D2	D1	D0
RGB565	First	R4	R3	R2	R1	R0	G5	G4	G3
	Second	G2	G1	G0	B4	B2	B2	B1	B0
RGB555	First	X	R4	R3	R2	R1	R0	G4	G3
	Second	G2	G1	G0	B4	B2	B2	B1	B0

Figure 5 Colour Encoding on Pixel Data

Other formats include YUV where luminance and chrominance are used with greyscale, which uses a single channel and therefore can transfer pixel data much faster. These data formats can be chosen depending on the image sensor module itself and configured by the I^{2}C control bus. Having a camera that can be configured for greyscale would allow for faster transfer of the pixel data to the FPGA and hence faster DVS Data processing, and a faster framerate or “Spike” Rate.

As the transfer rate of each pixel is tied to the PCLK, for larger resolutions a larger number of PCLK Cycles per frame will be required. Therefore, for a higher framerate, the resolution and the frequency of the PCLK Should be increased. In general terms, the theoretical framerate attainable is a function of the PCLK Speed, the resolution of the frame and the number of bytes transferred representing a single pixel [32]:

$$Theoretical\ Frame\ Rate = \frac{Maximum\ supported\ PCLK}{Horizontal\ size * Vertical\ size * \frac{Bytes}{pixel}}$$

Equation 1

2.5.3 Camera Serial Interface (CSI/CSI-2)

CSI is a serial interface designed for optimizing data transfer speeds in one direction, it comprises of a physical layer by which the data lanes are configured, and an I^{2}C control bus. There are a few versions, (CSI, CSI-2 and CSI-3), however, CSI-2 is the most prevalent on the market [31]. The licensing and complex nature of this highly optimized interface makes it very hard to build manually, as interfacing with a commercially available CSI-2 Camera would require the physical layer (D-PHY) to be built. There are few FPGA devices that support CSI-2 Interfaces and D-PHY Directly on the chip, however these are expensive and few. [33]

The basic principles of the CSI Interface are very similar to that of CPI, however, the interface makes use of low power and high-speed modes to optimize data rates on the line through the physical layer. A basic diagram showing CSI is given below in Figure 6.

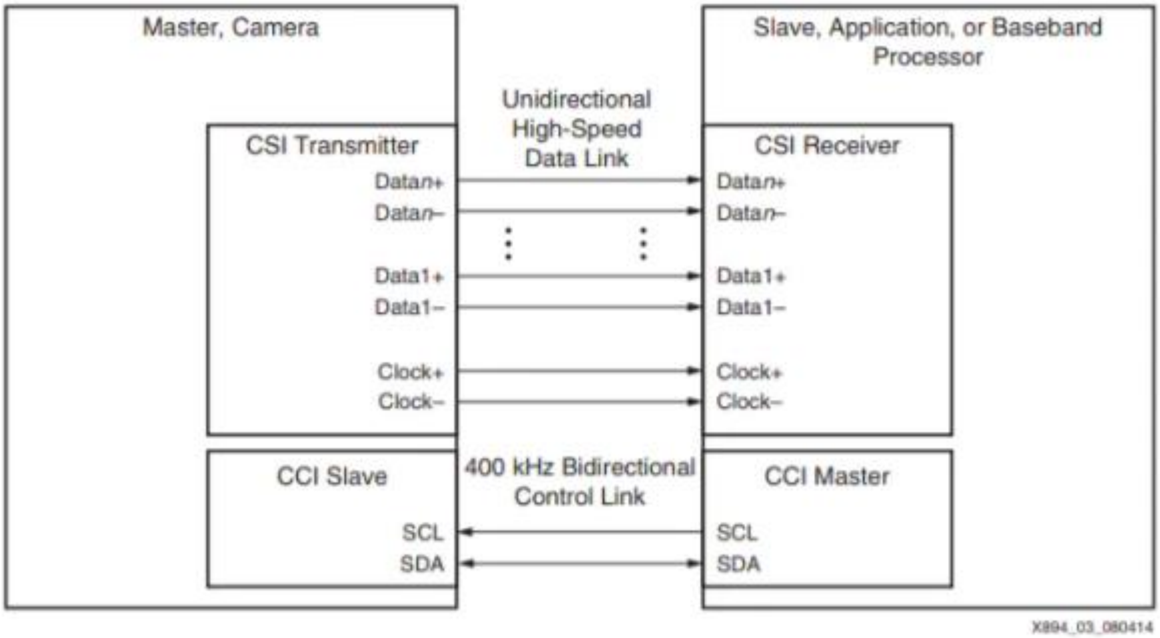


Figure 6 CSI-2 Basic Diagram [34]

With CPI, pixel data is sent on each rising edge of the Pixel Clock, however, with CSI-2, pixel data is sent on both edges. By using both edges, and up to 4 data lanes, speeds are achievable of up to 10GBps, far greater than that of the processing speed needed for the framerates of the ‘DVS’ Camera and FPGA demands. This is also configurable along with the data format used. [34]

The complexity of the D-PHY implementation and the timing requirements, together with the licensing of this standard mean that many CSI-2 products use an IP core, (a prebuilt configurable block that implements the system), which is also available for the PYNQ-Z2 as it is a Zynq device. The licensing for the CSI-2 Subsystem is available from Xilinx for a 4 month period for free, allowing for the testing of this method of interfacing with various commercial cameras and the FPGA used in this project. This provides a great platform for testing various camera solutions as many use this protocol.

2.5.4 Other Interfaces

There are other commercially used camera interfaces that could be used with the FPGA. MIPI DSI is another paid standard like CSI-2 and CPI, however, it uses differential signalling to send the pixel data serially [35]. Digital Video Port (DVP) is a standard parallel interface like CPI, however there is no standardised specification and it is usually implemented from scratch. HDMI input and USB input cameras could also be used, provided they have suitable Linux drivers that will work with the PYNQ Board. Camera Link, GigE and IEEE1394 are other methods that do not seem suitable for use with the PYNQ Board as their connectors are not present on the board or may be used for other things (Such as the Ethernet port for GigE). In addition, IP Cores are not readily available for use for these interfaces.

2.6 Machine Learning

Machine learning, which is a sub-discipline of artificial intelligence, involves creating computer programs that are able to learn from previous experiences in order to improve future performance [36]. This means that, instead of directly writing a program to solve a given problem, an engineer can create software that learns to perform the task autonomously. Machine learning has a wide range of applications in various different sectors, including finance [37], email classification [38], and cybersecurity [39], and is particularly useful in problems that are too complex for humans to perform. Various different machine learning techniques are required for this project, the relevant theory for which can be seen below.

2.6.1 Deep Learning

Deep learning is a machine learning technique which utilises deep neural networks (DNNs) to identify patterns in data such as images, video, or audio. DNNs are a type of artificial neural network (ANN), which are statistical models inspired by the structure of a human brain [40], and are made up of perceptrons, the function of which is comparable to biological neurons. Rather than using a custom-built algorithm for a specific application, DNNs are generic models which are trained for the intended purpose using large amounts of data. The way in which DNNs are structured, trained, and operate is described below.

2.6.1.1 The Perceptron

The fundamental building block of artificial neural networks is the perceptron: a relatively simple algorithm consisting of a number of inputs, a corresponding weight for each input, a single activation function, and a single output value. A block diagram showing the function of a perceptron can be seen below in Figure 7:

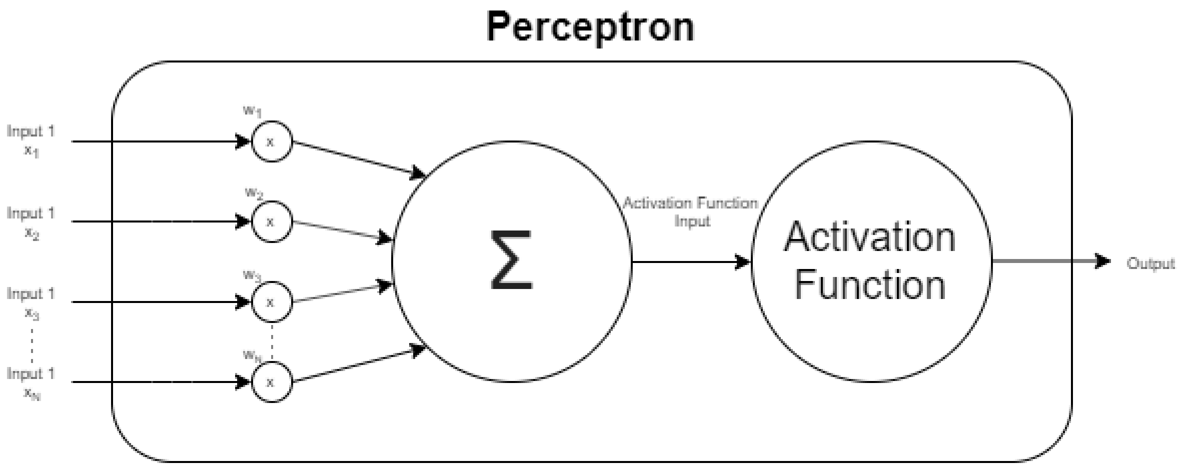


Figure 7: Perceptron Block Diagram

The output is calculated by first multiplying each input by its associated weight, and then adding all of these values together to find the activation function input. This can be seen algebraically below in Equation 2:

$$activation\ function\ input = \sum_{n=1}^{N} x_n w_n$$

Equation 2

This value can then be used as the input to the activation function, which is the function that determines the output of the perceptron. Various different activation functions exist, examples of which can be seen below:

Linear Activation Function

$$f(x) = x$$

Step Activation Function

$$f(x) = 0, if\ x\ < 0$$

$$= 1, if\ x \geq\ 0$$

Sigmoid Activation Function

$$f(x) = \frac{1}{1 + e^{-x}}$$

ReLU Activation Function

$$f(x) = 0, if\ x\ < 0$$

$$= x, if\ x \geq\ 0$$

Figure 8: Common Activation Functions

These activation functions can be seen graphically in Figure 9:

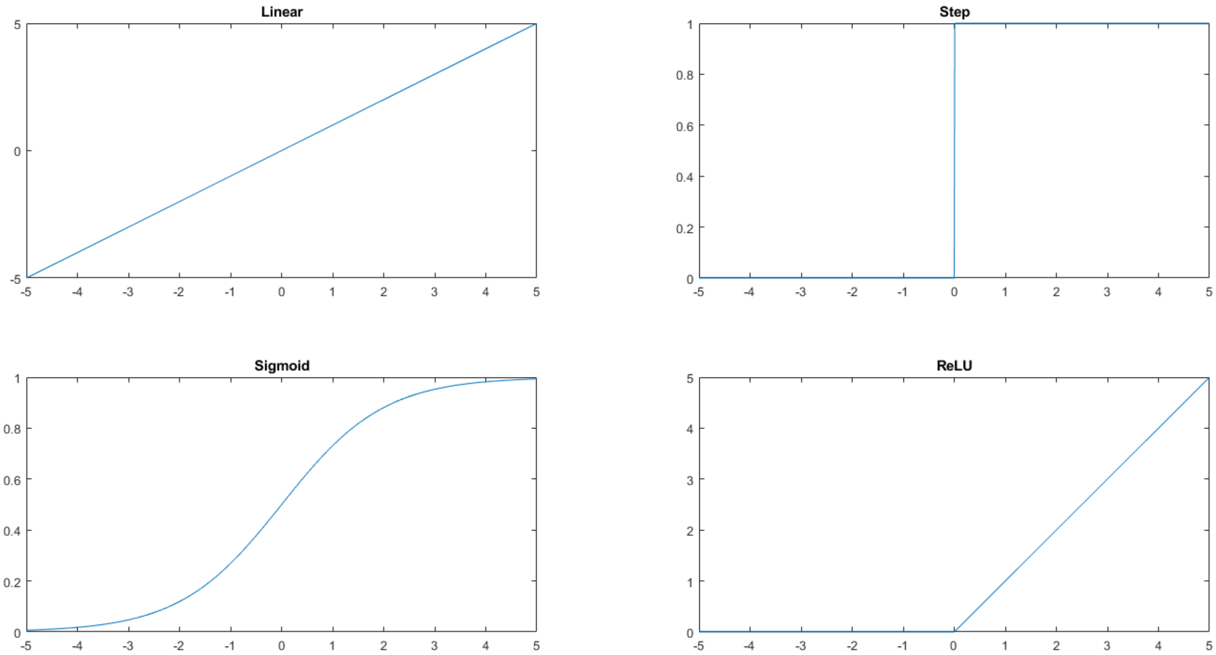


Figure 9: Plots of Common Activation Functions

The output of the activation then becomes the output of the perceptron, as seen in Figure 7. The reasons for choosing a specific activation function are discussed later. While perceptrons are not particularly useful on their own, they can be grouped to form ANNs.

2.6.1.2 Artificial Neural Networks

Perceptrons can be grouped into layers, which can then be combined into artificial neural networks (ANNs). A diagram of a very simple ANN can be seen in Figure 10:

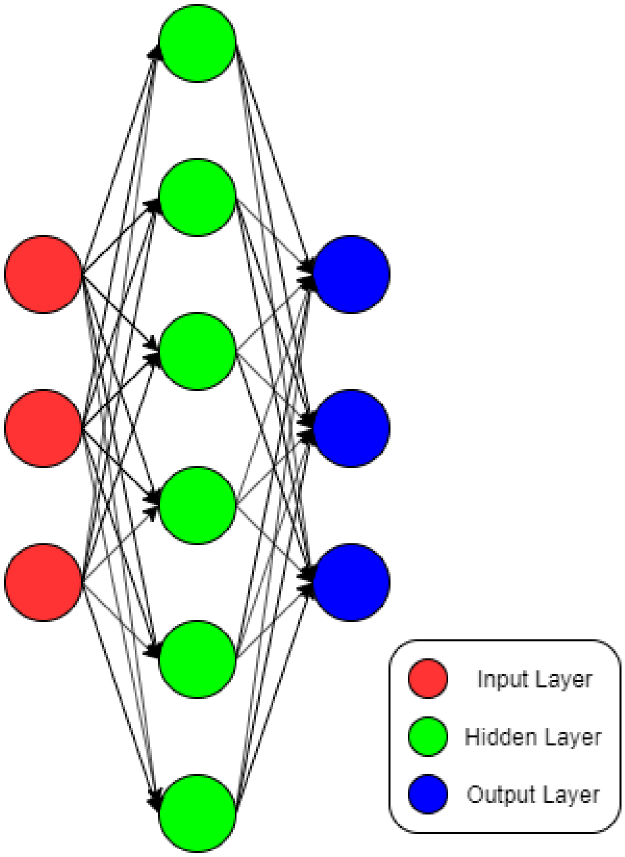


Figure 10: Diagram of Artificial Neural Network

We can see from Figure 10 that the multiple perceptrons have been grouped to form layers, and each perceptron is connected to all the perceptrons in adjacent layers. The layers are also categorised into three groups: input layer; hidden layer; and output layer. The input layer is used to feed data into the network, and the network's classification will be indication will be displayed on the output layer [41]. The hidden layer is defined as a layer that is neither the input layer nor output layer. ANNs similar to the one shown in Figure 10 can be used to detect simple patterns in various different types of data, but struggle with more complex tasks. This lead to the development of Deep Neural Networks (DNNs).

2.6.1.3 Deep Neural Networks

A DNN is defined as an artificial neural network with more than one hidden layer [42]. The additional hidden layers allow the network to identify more sophisticated features. Several different types of DNN exist, with one of the most versatile types being convolutional neural networks (CNNs). CNNs are capable of detecting patterns in various different types of data, particularly visual data. They use convolutional and pooling layers to identify various features in data [43].

The convolutional layers are able to extract high level features, such as edges, by using convolutional kernels. These layers can be combined to detect more complex patterns, such as shapes, letters, or faces. Pooling layers are used to decrease the number of pixels in an image to decrease the computational cost of the network. When these layers are combined with fully connected hidden layers, as shown in Figure 10, a CNN is able to extract very sophisticated features from data.

Deep Neural Networks, unlike non machine learning techniques, require large amounts of data and a training algorithm [44]. The training algorithm requires a loss function, which measures how accurate the networks output is. When the DNN is first made, each perceptron is assigned random weights, and then data is fed into the network and the loss function is calculated. The training algorithm then varies the weights of the network so that the loss function is minimised. This process is repeated until the network can consistently produce correct outputs.

2.6.2 Voice Recognition

2.6.2.1 An Introduction to Voice Recognition

Voice recognition, also known as speech recognition, is the practice of using software and/or hardware to process human voice in order to perform analysis such as the recognition of commands or distinguishing between speakers [45]. It is used in many applications such as speech-to-text software, dictation in smartphones [46] [47] , and automated telephone systems to interact with callers and direct calls [48].

Voice recognition is commonly based on phonemes. Phonemes are the smallest element of a language i.e. the smallest sound it’s possible to make such as the “puh” sound from the letter p. Usually, the first stage of recognition is to break the input words down into these phonemes. Then, they are analysed based on their position in the word or phrase, and a model can be used to predict what has been said [49].

The four main methods to classify speech are simple audio pattern matching, complex pattern analysis, statistical analysis, and artificial neural networks [50].

Simple pattern matching involves comparing the frequencies present in the speech signal to a known template. This works extremely well for a small vocabulary, but it quickly becomes inefficient as the size of the vocabulary increases. On the other hand, complex pattern matching aims to learn the basic building blocks of words and use that to understand speech, however, this is not very accurate [50].

Statistical models use probabilities to determine how likely it is that one phoneme follows another and how likely it is that one word will to follow another. As such, different statistical models are required for different languages. In particular, a statistical model called a Hidden Markov Model has been shown to work very well and is the favoured approach by many [50], [51], [52].

Finally, deep learning learns features from a training set of words. These can then be used to determine what has been said. Although it is a very popular method of recognizing speech in recent times, being explored by large companies like Google and Microsoft [50], it, unlike a statistical model which has a view of the whole language, a deep neural network knows only the features that it has been trained on. Depending on the application, this can be helpful or a hindrance. It also requires a large amount of training data and has a high computational cost. However, for the correct applications, such as the Google Speech API [53], this can translate to a very high accuracy [54].

There is no right or wrong answer as to which of these approaches is best; it depends largely on the application [50], [55].

2.6.2.2 Hidden Markov Models

Hidden Markov Models recognise speech using phonemes and the probabilities of those phonemes occurring in a certain order. A word is considered a “chain” and each phoneme is a link in that chain [49]. An iterative process is used to reconstruct the word by breaking down the phonemes that the system thinks it has heard and determining how likely it is that one phoneme follows another. This is best explained by example. Say the word the word to be detected is “dolphin”, the steps in the recognition would be as follows [56]:

Stage 1: Identify the first phoneme. In this case, the system thinks it could be a D (0.9 probability), a T (0.4 probability) or a G (0.6 probability). Since D seems the most likely for what was heard, it is taken forward to the next step.

Stage 2: How likely is the combination of phonemes that was heard? The next phoneme the system thought it heard was “o”. In the English language, an “o” following a “d” is reasonably probable, so it checks the next phoneme, which is “l”. Again, in English, an “l” following an “o” is not unreasonable, so it continues. This stage repeats until it thinks it has a completed word. If at any point, the system hits a phoneme combination that is not valid in the model language, then it will go back to and reassess what it has heard and go down a different branch.

Stage 3: How likely is it that what was said was truly the word built in stage 2, based on the context? For example, stage 2 might provide us with two separate words “doll” and “fin”, but when it gets to this stage it realises that these two words don’t often follow each other and goes back to stage 2 to reassess in order to realise that, in fact, it is only one word: dolphin. The stage can also use grammar to determine the correctness of its guess.

Stages 2 and 3 highlights the need for different statistical models for different languages. A common letter combination in one language may not exist in another. In fact, not all languages use the same phonemes. For example, there is no "z" sound in the Welsh language [57]. Similarly, grammar and sentence structure varies for different languages. The models are trained on the probabilities of each phoneme in a language, probabilities of phonemes following each other, and probabilities of words following each other. There is no need to train a model for this project as it is possible to download an existing model for the English language for many different environments, including Python [58].

2.6.2.3 Where Voice Recognition Struggles

Voice recognition naturally struggles with homophones (words that sound the same) such as they’re, their, and there. However, in these cases, the correct word can be found by looking at the context [49]. Things like background noise, background conversation (“overlapping speech”), and noise from a poor microphone are all much bigger issues that impact on voice recognition [49].

Filtering can be used to reduce some of this noise, however, it does so at the expense of the quality of the desired voice signal. In order to improve the quality of the recorded audio, the best approach is to have a good quality microphone, with a good preamp and perhaps some noise-cancelling qualities, to ensure that the signal is as clean as possible to begin with [59].

3 Procedure

3.1 Project Planning

The project brief was analysed, and a system design consisting of six parts was formulated. The system is made up of a passive motion sensor, a camera module, an FPGA to create the DVS images, a wireless transmission sub-system, an image classification sub-system, and a subsystem to communicate with the user.

The camera will record data and send data to the FPGA, which will subtract subsequent frames in order to approximate DVS data. The data will only be transmitted to the computer when a passive sensor has detected motion. This is to save power, but also to put the user at ease because the recordings are not all being saved. The fall classification system will then receive this data and determine whether the detected motion was a fall or not. If it was a fall, it will trigger the voice recognition system. This part of the system will talk with the user to determine what kind of help they need: an ambulance, a friend, or no help. Depending on the answer, the system will call the appropriate help or turn off. A high-level system diagram was then made, which can be seen in Figure 11.

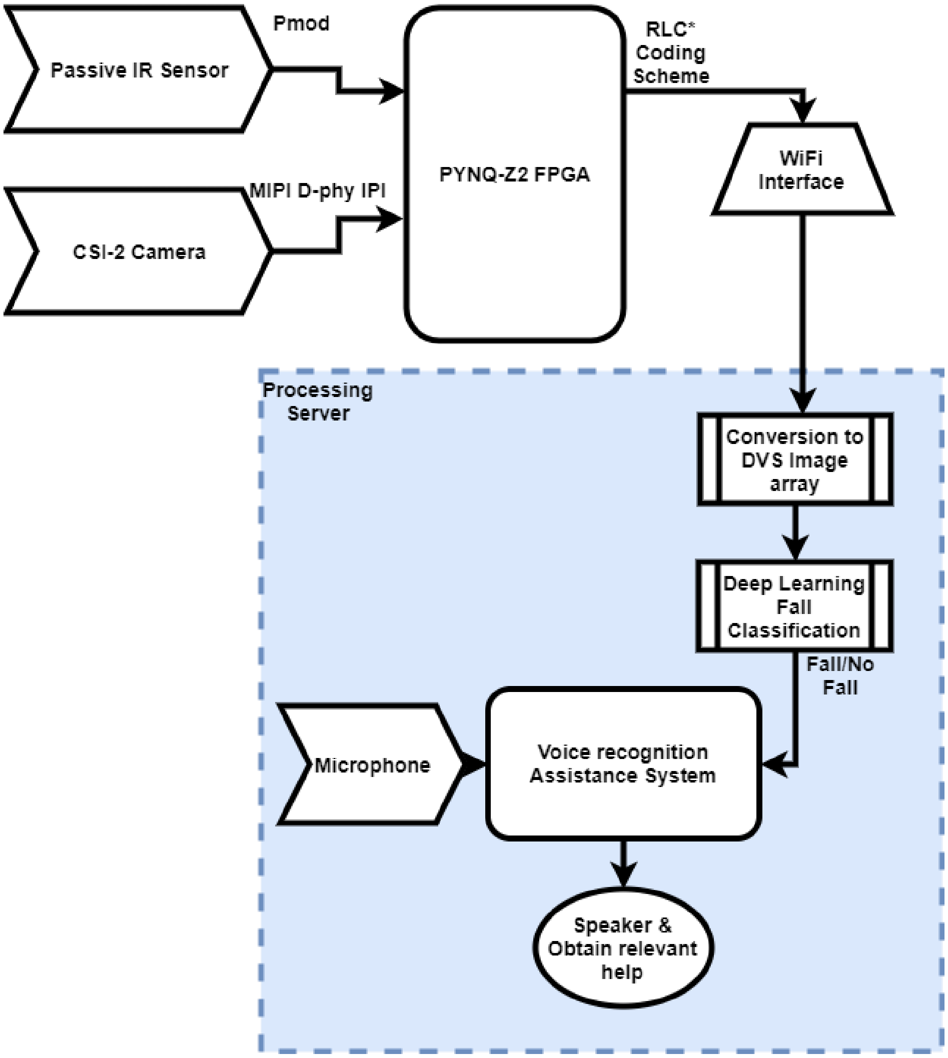


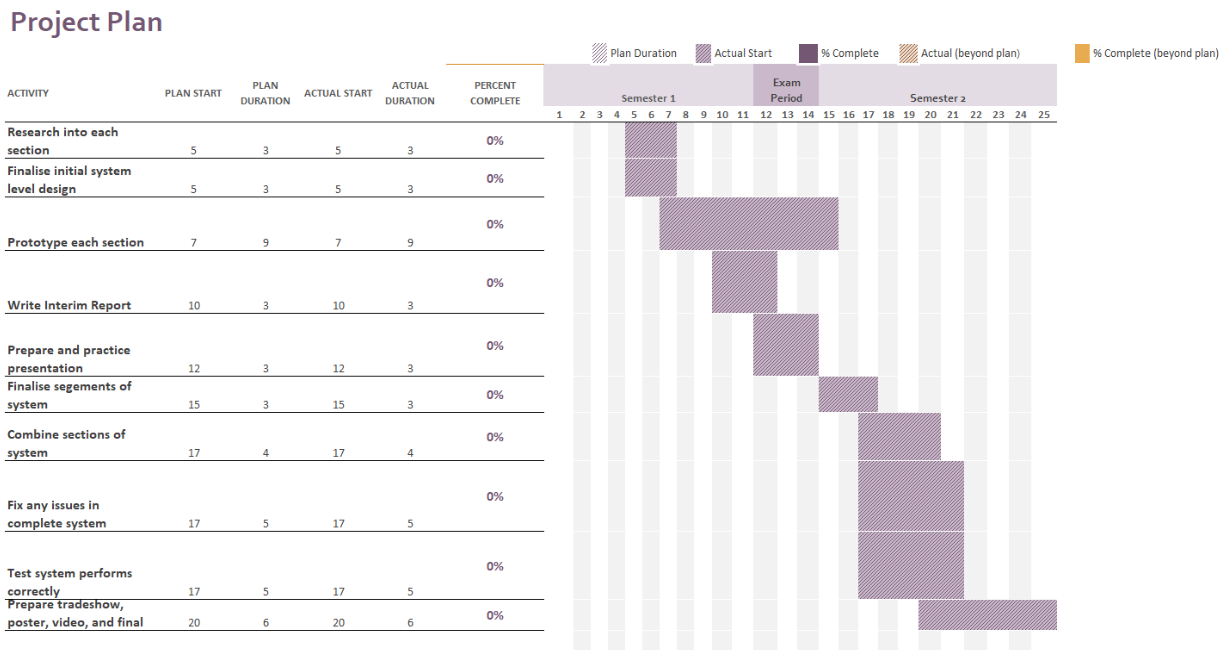
Figure 11: System-Level Designs

The project was then split into objectives, and each objective was given a level of importance. These can be seen below in Table 1:

Table 1: Project Objectives

Project Objectives	Importance
Gain appropriate background knowledge of the problem, existing technologies, and possible solutions	Major
Develop FPGA/Camera system to replicate a DVS camera using a low-cost sensor	Major
Develop a method of detecting activity within the room, so that the system knows when to transmit data	Major
Detect activity in the room using a passive sensor	Optional
Transmit DVS data wirelessly securely to computer to be processed	Major
Use data from DVS camera to determine if a person has fallen	Major
Use a speaker and microphone to speak to the person, and determine if emergency services are required	Major
Allow the person to choose who will be contacted e.g. emergency services, relative, neighbour	Minor

These objectives were then assigned an importance, and a Gantt Chart was created to ensure that the project stayed on schedule. This can be seen below in Figure 12: Initial Gantt Chart:



Project Plan

ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Research into each section	5	3	5	3	0%
Finalise initial system level design	5	3	5	3	0%
Prototype each section	7	9	7	9	0%
Write Interim Report	10	3	10	3	0%
Prepare and practice presentation	12	3	12	3	0%
Finalise segements of system	15	3	15	3	0%
Combine sections of system	17	4	17	4	0%
Fix any issues in complete system	17	5	17	5	0%
Test system performs correctly	17	5	17	5	0%
Prepare tradeshow, poster, video, and final	20	6	20	6	0%

Figure 12: Initial Gantt Chart

In order to minimise the impact of unforeseen delays and issues, some possible future risks were identified, and mitigating action was identified. This can be seen in Table 2:

Table 2: Possible Risks and Mitigating Actions

	Possible Risk:	**Mitigating Action:**
1	Sickness	Clear contact with team, proposer, and mentor if work progress may be compromised
2	Loss of work	Constant back-up of work using services like the university H drive, GitHub, and USB devices
3	Hardware break/failure	New equipment acquired or old equipment repaired
4	Project too ambitious/learning curve too high	Scope reduced

The group intends to meet internally weekly to review progress and set intermittent goals. Additional meetings with the project proposer and project mentor will be arranged as required.

3.1.1 Camera Interface

In terms of choice for the camera interface with the FPGA, there are two clear candidates: the “Camera Parallel Interface” (CPI) and the “Camera Serial Interface” (CSI). Both are easily implemented on the FPGA and provide sufficient data rates for processing the image data at a high framerate dependent on the camera module itself and its configuration. [60] In implementing the “Low-Cost DVS” the framerate of the camera is a particularly important feature as the speed at which two frames can be read into the FPGA and processed into ‘DVS’ Data is directly dependent on how fast each image is captured and transferred to the FPGA for processing.

CPI starts to break down at much higher speeds of operation where the micro-resistances from differences in track lengths can throw off the timing of the data signals, so CSI would be more robust. [61] However, it is unlikely that we will require such high speeds without purchasing an expensive camera, so the choice of interface will primarily be decided by the price and performance of the camera module itself. By use of the IP Core, CSI may also be simpler to implement on the FPGA, as it is a premade configurable block, so this is preferred.

3.1.2 Wireless Transmission

The transmission of the video feed from the DVS camera to the server can be done through a wired or wireless medium. A wired connection would mean that the DVS camera could only be installed next to a PC, which might greatly limit its field-of-view (FOV). Therefore, a wireless implementation is more suitable for the fall detection system, as it can be placed anywhere in the room where the FOV of the camera is maximized.

3.1.3 FPGA

An FPGA, particularly a SoC device, is a sensible choice for the “low-cost DVS” aspect of the project due to the processing capabilities within a small, light-weight board. This allows the FPGA and a camera to be mounted as one object on a wall or similar whilst also processing the captured images to create the desired emulated DVS events without using a much a larger computer.

The PYNQ-Z2 board was a natural fit as the FPGA of choice as this allowed access to popular and powerful Python image processing libraries such as “openCV” [62]. This will save vital time on the project as many useful functions are available on-demand, allowing more time to be spent on other areas of the project.

3.1.4 Python

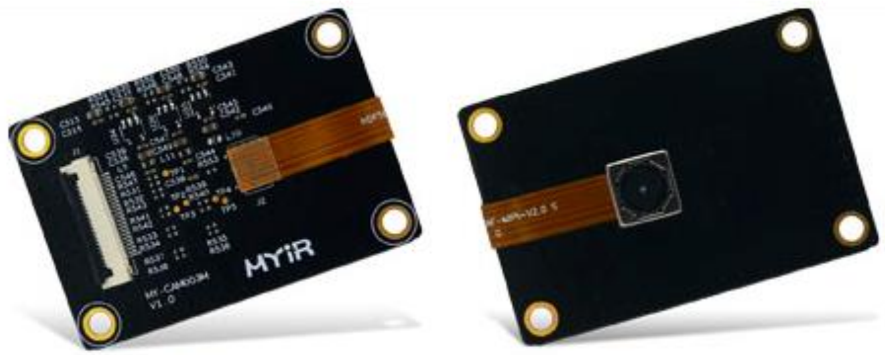
The PYNQ framework uses the Python programming language. Python is a popular language for deep learning and machine learning thanks to its simplicity and large number of free, useful libraries such as SciPy, SpeechRecognition, and TensorFlow [63]. Additionally, it’s desirable that the whole project uses the same language to ensure compatibility between parts. Therefore, Python was chosen for this project.

3.1.5 Voice Recognition

This project has a vocabulary of only two words (“yes” and “no”) that might later be extended to include at maximum a few more words like “stop”. This makes the simple pattern matching approach desirable for this project as opposed to a deep learning approach, because deep learning not only takes a large amount of resources but also would be inflexible to adding new words into the vocabulary without additional training. Statistical models have been one of the most accurate approaches for a long time, and models for the English language are readily available. Therefore, for this project, a joint approach of simple pattern matching and statistical modelling is used. In particular, a Hidden Markov Model is used.

3.2 Camera Module

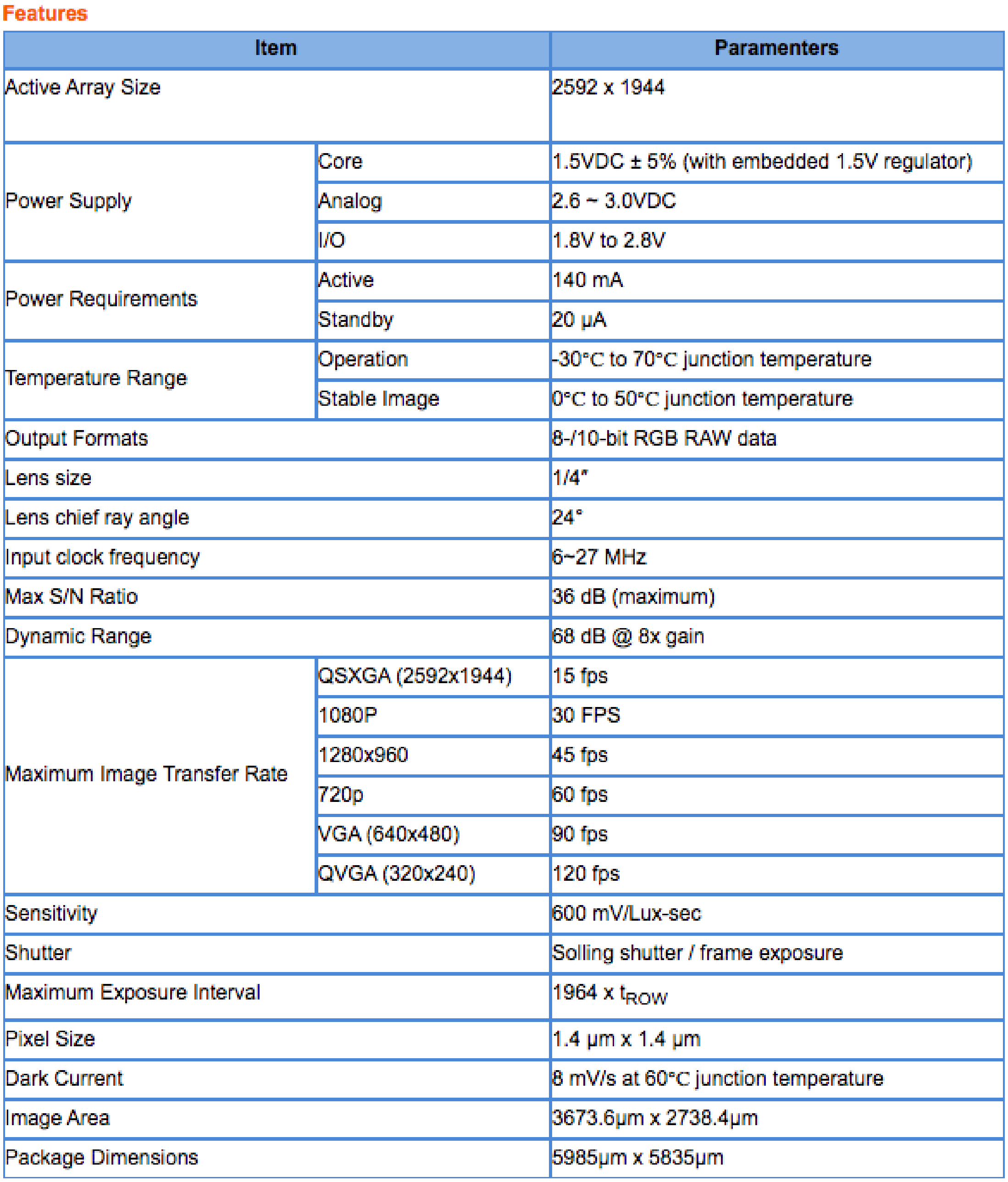
The Camera Module used is the MYCAM003M, from MYIR. A picture of the device is shown below in Figure 13.



MY-CAM003M MIPI Camera Module (top-view and bottom-view)

Figure 13 Picture of Target Camera Module

The capabilities of the device are shown in the device specifications below in Figure 14.



Features

Item		Paramenters
Active Array Size		2592 x 1944
Power Supply	Core	1.5VDC ± 5% (with embedded 1.5V regulator)
	Analog	2.6 ~ 3.0VDC
	I/O	1.8V to 2.8V
Power Requirements	Active	140 mA
	Standby	20 µA
Temperature Range	Operation	-30°C to 70°C junction temperature
	Stable Image	0°C to 50°C junction temperature
Output Formats		8-/10-bit RGB RAW data
Lens size		1/4"
Lens chief ray angle		24°
Input clock frequency		6~27 MHz
Max S/N Ratio		36 dB (maximum)
Dynamic Range		68 dB @ 8x gain
Maximum Image Transfer Rate	QSXGA (2592x1944)	15 fps
	1080P	30 FPS
	1280x960	45 fps
	720p	60 fps
	VGA (640x480)	90 fps
	QVGA (320x240)	120 fps
Sensitivity		600 mV/Lux-sec
Shutter		Solling shutter / frame exposure
Maximum Exposure Interval		1964 x t_{ROW}
Pixel Size		1.4 µm x 1.4 µm
Dark Current		8 mV/s at 60°C junction temperature
Image Area		3673.6µm x 2738.4µm
Package Dimensions		5985µm x 5835µm

Figure 14 Camera Module Specifications

Ideally, we would like to maximize fps while keeping at minimum a standard quality resolution to improve the performance of the DVS data processing. Obtaining an fps higher than 60 would improve upon the basic method of using an usb2.0 or HDMI Camera. This camera is flexible and can obtain 90fps when configured at 640p. It is also cheap and can be used to prove the interface works, then replaced with a slightly more expensive 120fps camera using the same interface.

The camera module was designed for use with the MYD-JX8MX development board, in which it is connected via a 24 Pin 0.5mm FPC Connector via ribbon cable. There is no datasheet readily available for the device but there are device notes for the module on the MYIR Site pertaining to its use with the MYD-JX8MX board.

This was used to confirm the connector details and to source an adapter that would take the 24pin FPC Connection and break the pins out to more usable through the hole DIP (Dual In-line package) Connections. These DIP sockets can be used to directly connect to pins on the PYNQ Board, provided a pinout is known for the device. This adapter is shown connected to the camera module on the left in Figure 15.

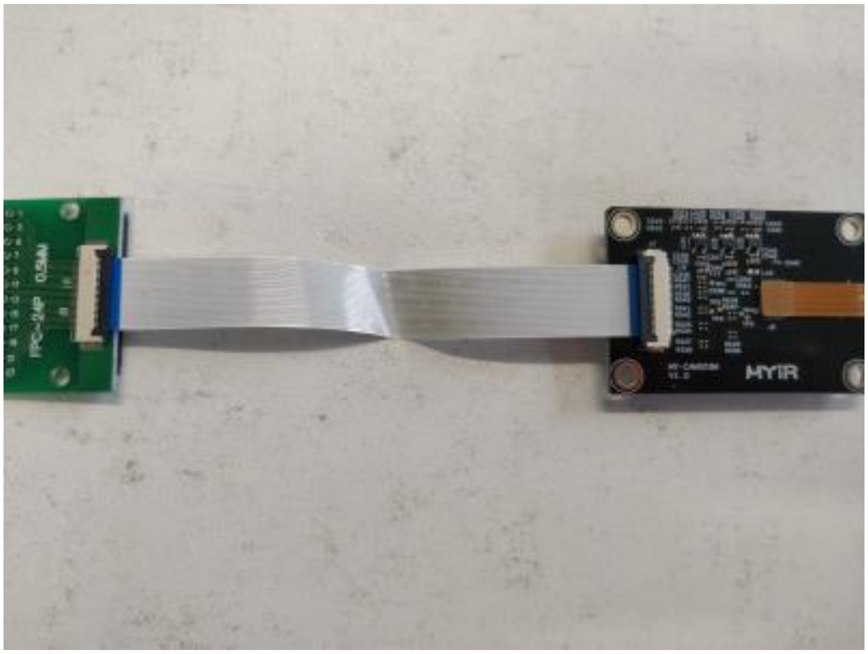
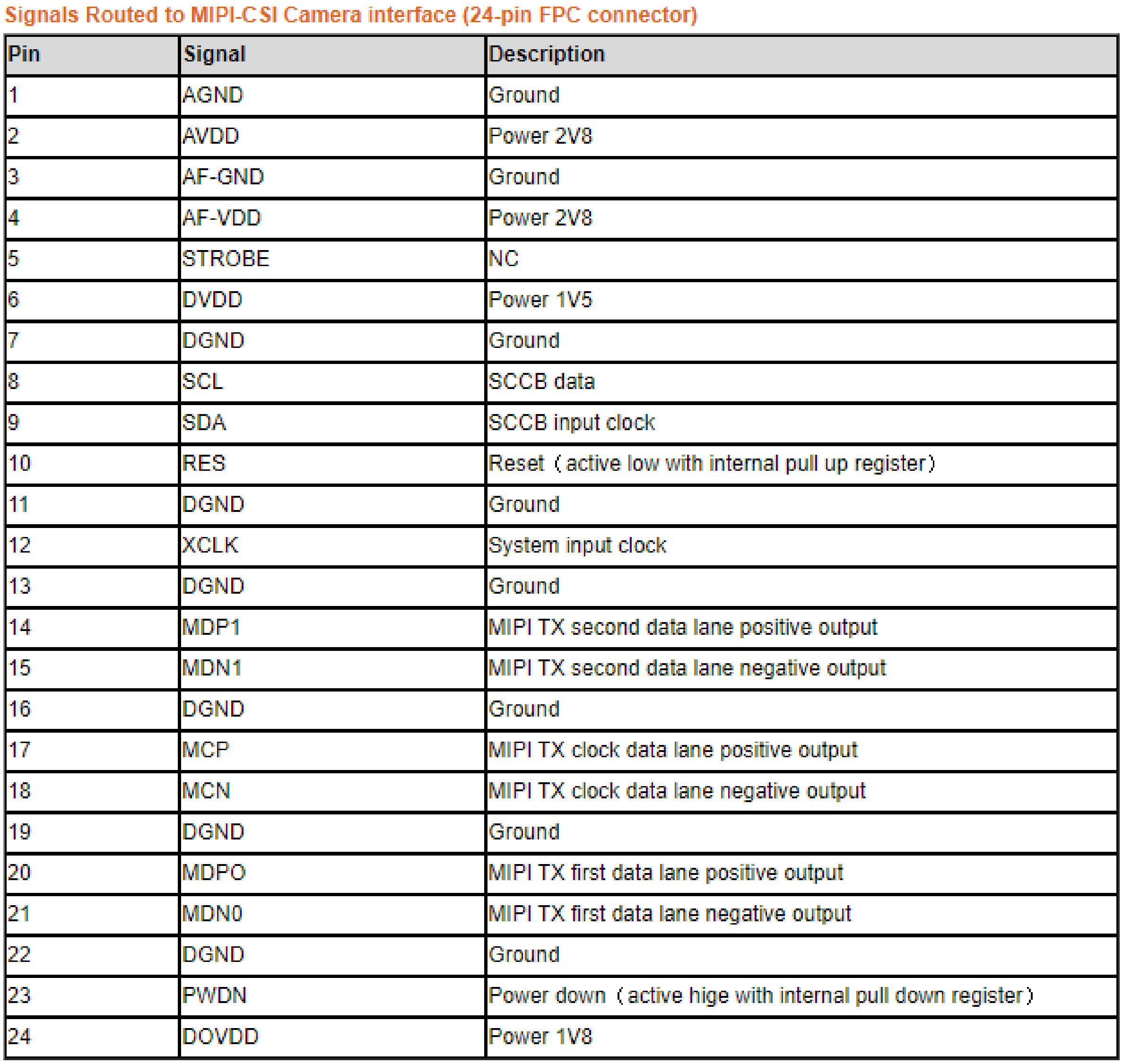


Figure 15 MYIR Camera to 24 FPC Connector

This pinout can be found on the camera module page on the MYIR Website and is shown below in Figure 16.



Signals Routed to MIPI-CSI Camera interface (24-pin FPC connector)

Pin	Signal	Description
1	AGND	Ground
2	AVDD	Power 2V8
3	AF-GND	Ground
4	AF-VDD	Power 2V8
5	STROBE	NC
6	DVDD	Power 1V5
7	DGND	Ground
8	SCL	SCCB data
9	SDA	SCCB input clock
10	RES	Reset（active low with internal pull up register）
11	DGND	Ground
12	XCLK	System input clock
13	DGND	Ground
14	MDP1	MIPI TX second data lane positive output
15	MDN1	MIPI TX second data lane negative output
16	DGND	Ground
17	MCP	MIPI TX clock data lane positive output
18	MCN	MIPI TX clock data lane negative output
19	DGND	Ground
20	MDPO	MIPI TX first data lane positive output
21	MDN0	MIPI TX first data lane negative output
22	DGND	Ground
23	PWDN	Power down（active hige with internal pull down register）
24	DOVDD	Power 1V8

Figure 16 CSI-2 Interface Breakout Pin Routing

By using this pinout, the camera can be interfaced with through configured pins on the PYNQ Board, providing a stable connection for images to be transferred.

3.3 FPGA Design

The FPGA design is the cornerstone of the “low-cost DVS” aspect of the project. This design must be able to receive input image frames from a traditional camera, process the differences between these frames and output events to emulate a true DVS camera. These emulated events must be in a form which can be wirelessly transmitted.

3.3.1 Initial IP Integrator Focused Methodology

The method of implementing this design has gone through a number of iterations as the understanding of the PYNQ design flow has increased. The author has come from a position of using IP Integrator within Vivado as the approach to implementing an FPGA design on an SoC device. This therefore, was the natural starting point of creating the design for this project, leading to an exploration of hardware solutions which would allow the input of RAW camera data to the system.

Based on the camera research it was decided that that the chosen IP block must be MIPI based and support a camera using CSI-2. This led to a natural fit with the MIPI CSI-2 Receiver Subsystem [64], shown in Figure 17. This IP block is a subsystem consisting of several sub-cores, up to 4 D-PHY lanes, support for RAW, RGB and YUV data types as well as AXI I^2C support. All of this makes the subsystem ideal for this project as the camera data can be translated to an AXI stream which can be manipulated through the rest of the design.

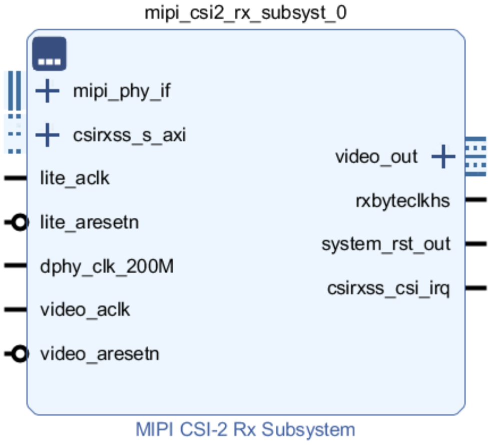


Figure 17: MIPI CSI-2 Receiver Subsystem

These sub-cores are shown in the system architecture overview in Figure 18. The MIPI D-PHY layer provides physical layer protocols and is the high-speed interface between the camera data and the FPGA PL. The MIPI CSI-2 Rx Controller acts as the ‘brains’ of the core, controlling lane management of the data and low-level protocols abstracting these from the user. The AXI Crossbar is an address manager, routing AXI logic to the correct sub-cores. The Video Format Bridge filters the input data so that only the data required by the user is output to the rest of the design. This block also processes the input data into the required pixel packing data type. Finally, the AXI IIC interface is optional if the user requests it allows much faster output from the IP if the design can support it.

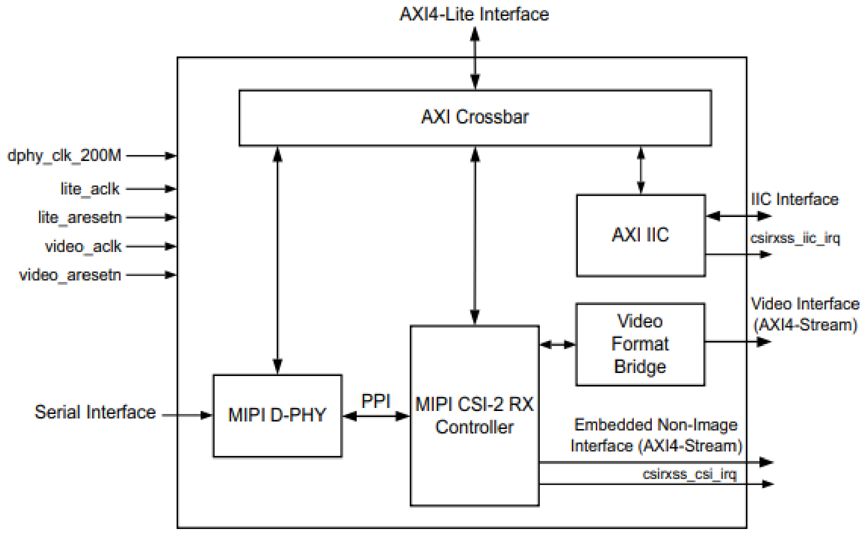


Figure 18: System Overview for MIPI CSI-2 Receiver Subsystem [64]

Provided with this core is an example design for the ZCU102 FPGA, shown in Figure 19 . This block diagram is too large to quickly grasp, however, it is implementing rather straightforward functionality.

First, data is read from a connected camera sensor and converted to AXI stream data using the MIPI CSI-2 Receiver Subsystem. This raw camera data is converted into RGB data through an image pipeline; a process of demosaicing, interpolating the raw data into a matrix of coloured pixels, followed by a Gamma Lookup Table to perform gamma compression - a method of optimizing bit usage within still images. This is followed by two IP cores to perform video processing from the still images (the video processing subsystem) and to directly interface between video and memory access (VDMA). Finally, there are two output methods from the system, a direct HDMI Out to HDMI Monitor and a MIPI transmitter to a DSI Display Panel that could be found in a mobile phone or a laptop [64].

To put this simply, the model inputs still RAW image data from a camera, converts to a standard image format, converts this to video data, and outputs the video to a screen.

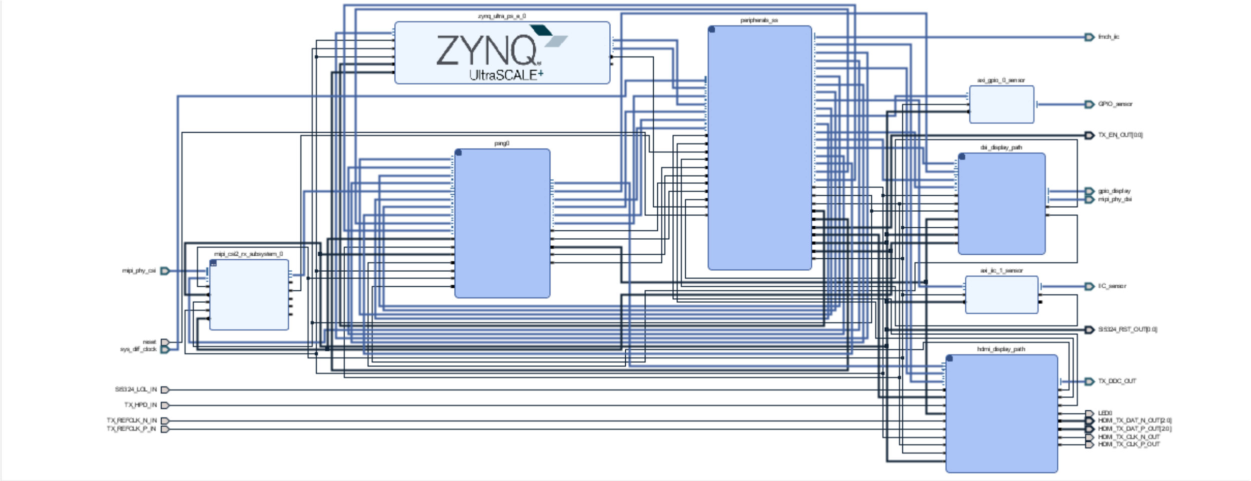


Figure 19: ZCU102 example design

This model at first seems like the ideal basis for the DVS design, swapping out the UltraScale PS and the soft MicroBlaze processor for the required Zynq-7000 PS. The design could be used to test that data is being successfully read from a camera and processed through a system before editing the design to allow for DVS emulation. This however, is the issue; the DVS element is merely a second thought with this method, not utilizing the power of the PYNQ software.

3.3.2 Exploration of PYNQ and the Base Overlay

With this in mind, attention was turned to the PYNQ documentation and the surprisingly detailed PYNQ tutorials, provided through the PYNQ workshop GitHub [65]. These tutorials begin simple, controlling on-board LEDs, switches and buttons through Python code [66]. However, the idea of utilizing the PYNQ overlay was quickly introduced using the pre-installed base overlay as a powerful tool to quickly implement a desired application in both PS and PL [67]. These overlays are a clearly superior choice to a standard IP Integrator design, allowing any PS functionality to be easily incorporated onto a previous PL design.

This leads to a potential solution to prove that the method is successful. There are example notebooks available for video to the PYNQ-Z2 using a USB [68] or a HDMI camera [24], such as a standalone webcam or a laptop. These cameras are limited but provide a means of testing the system whilst maintaining a focus on Python code as the major focus. Following this, the DVS code can be incorporated without making any changes to the overlay itself, simply implementing the process in Python onto the tested design.

3.3.3 Combined Design

The base overlay, whilst versatile is still limited in its functionality. This is where custom overlays become useful, either designed from the base overlay or from scratch, allowing both the PL and PS aspects of the design to be customized by the user. For this project it may be desirable to use the block design used to create the base overlay and add required IP to allow RAW image data to be input to the system bringing the previously discussed MIPI CSI-2 Receiver Subsystem IP back into the design. Now instead of building a design based on this IP's example design, the IP will be inserted into the base design, along with any required image pipeline, to enhance the functionality of the overlay.

3.3.4 DVS Emulation

With PS and PL easily controlled by an overlay loaded straight to the board, it becomes much more straightforward to implement DVS emulation. The overlay can be input to a Jupyter Notebook, allowing standardized Python image processing libraries to be used to assist in the image differencing, to create the events that would be expected as an output from a DVS system.

Python may also offer a solution for the encoding of the DVS data for transmission. Alternatively, this may be approached from a PL solution, where another IP could be added to the custom overlay allowing the data encoding to be abstracted from the user.

3.4 Wireless DVS camera system

An important design requirement of the fall detection system is portability. This means that the DVS video feed must be streamed from the PYNQ-Z2 FPGA to the processing server. This is achieved through a wireless communications system. Such a system is vital as it allows the DVS camera to be installed in a place that maximizes the FOV of the camera. An optional criterion of this system is to be a half-duplex system, which means that data can be transmitted in both directions but not at the same time. Another ideal feature of the communications system is to be able to remotely control the SoC FPGA. The last optional criterion is to have a system that supports multiple DVS cameras on the same network. In the beginning, several popular wireless transmission systems were investigated, such as: Bluetooth, SDR and Wi-Fi (WLAN).

3.4.1 Bluetooth

Bluetooth refers to the wireless standard which uses radio waves to transmit data at ~2.45 GHz. It is a packet-based protocol with a master/slave architecture. This technology does not meet the requirements of the communications system as it has a low data transmission rate of 2.1 Mbps, due to the low bandwidth available of 800 Kbps [69]. Bluetooth cannot have a seamless transition between transmitting the video feed to the processing system and using the processing system to remotely control the FPGA because of the master/slave architecture. The connection between the two devices would have to be reinstantiated to change between the master and slave. Another downside of Bluetooth is the limited range of 5-30 meters. The quality of the connection between the DVS camera and the processing system would be very bad if these two are found in different rooms.

3.4.2 Software Defined Radio

Initially Software Defined Radio seemed like the communications system which was most likely to be used. It can offer high data transmission rates if the transmitter has a high data throughput, and it can also transmit information over any arbitrary frequency as long as it is legal to do so. This makes it possible to “hide” the network as the receiver must know the frequency over which data is being sent, which leads to having a very secure network. The SDR could be also used to receive data, if a receiver was built. This receiver could have been used as a “radar” which detected noise in the radio spectrum that was caused by people moving in the room where the DVS system is deployed. When the “radar” picked-up movement it would then start to stream the video feed from the DVS camera to the processing server.

However, several major drawbacks were found after more research in the SDR field. Additional peripherals, such as antennas, are required to be able to transmit and receive data via RF. However, PYNQ does not have SMA connectors integrated on the board. An SMA connector is a coaxial RF connector, and it can be seen in Figure 20 below. Moreover, the PYNQ board does not have a high-speed ADC/DAC integrated which are required to implement the SDR.

Figure 20 SMA Connector (Male)

An ADC Pmod and a DAC Pmod were found available on the Digilent website. These modules also have SMA connectors integrated in them. However, the ADC was created to read voltage measurements and it cannot listen to high frequency radio transmissions due to its small sampling frequency of 4.8 KHz. According to the Nyquist theorem, the receiver could only reach signals up to 2.4 KHz, which is not ideal for this SDR. These peripherals also have a total cost of ~£50 [70] [71].

The next possible SDR implementation was found to be the ADALM-PLUTO active learning module, developed by Analog Devices. This module can be seen in Figure 21 below. This device is a self-contained SDR and its internal hardware architecture can be seen in Figure 22.

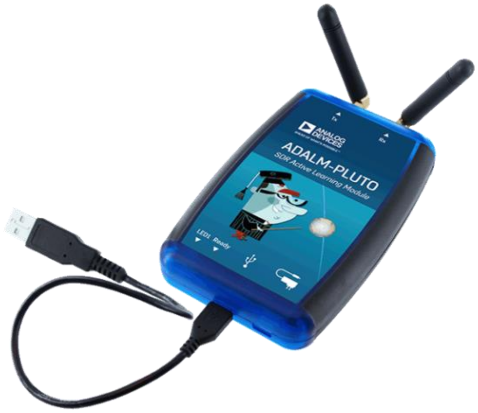


Figure 21 ADALM-PlutoSDR Learning Module [72]

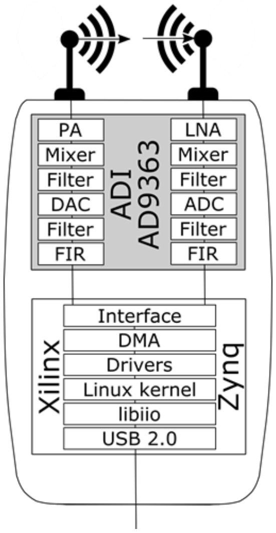


Figure 22 ADALM-Pluto Hardware Overview [72]

As Figure 22 shows, this module contains the AD9363--Highly Integrated RF Agile Transceiver and Xilinx® Zynq Z-7010 FPGA. It can be used via the USB 2.0 interface. This device is also capable of transmitting and receiving data, with a bandwidth of 20 MHz and it has an RF coverage from 325 MHz to 3.8 GHz [72]. Unfortunately, even if this device can be connected to the USB port on the PYNQ-Z2 board, it does not have a driver for it. This driver would facilitate the communication between the Zynq PS on the SDR module and the Zynq PS on the PYNQ-Z2 board. Another drawback of this module is its cost of ~£110, as it is desired to create a low-cost DVS fall detection system.

Therefore, the SDR communications system was proven to be unfeasible for this project as the PYNQ does not have the fundamental high-speed ADC/DAC components and the ADALM-Pluto self-contained SDR cannot be used through the PYNQ-Z2 board.

3.4.3 Wi-Fi (WLAN)

The last wireless technology investigated is Wi-Fi. As it was mentioned in the background section, Wi-Fi uses the 2.4 GHz frequency to communicate between devices. It has a bandwidth of 11Mbps, which enables fast data transmission [69]. Wi-Fi is also a half-duplex system and it can be used to create a Wireless Local Area Network, both of which meet the optional criteria in the system selection. Therefore, the PYNQ-Z2 could be remotely operated from any device that is connected to the same Wi-Fi network. It is also a secure network as long as it uses WPA2 PSK security protocol, which is the safest encryption method.

The Pmod Wi-Fi interface manufactured by Digilent seemed to be the solution to provide Wi-Fi access to the PYNQ-Z2. This peripheral also came with drivers required to connect it to the FPGA. However, after getting the component and building the HDL design for the Wi-Fi interface it was found out that the driver is incompatible with any Vivado versions older than 2017.4 (the latest version being 2019.2). The driver incompatibility is caused by a bug in the C compiler of the Xilinx Software Development Kit (SDK). Digilent knows about this issue but they have yet to allocate resources to fix it [73].

A different approach had to be taken to create a wireless system, since the Digilent peripheral proved to be unsuitable. The next appropriate Wi-Fi interface identified is the “Ralink 5370 Wi-Fi Dongle”, which can be seen in Figure 23 below. This peripheral can be connected to the USB slot on the PYNQ-Z2 board. This module can be used through the pre-existing PYNQ Wi-Fi module, which is a python module used to interact with Wi-Fi adapters [74]. This method offers full wireless access to the Jupyter Notebooks, which means that the ARM PS can be fully controlled remotely. Therefore, the board can be programmed from a PC or a smartphone or anything that has a browser and access to the same Wi-Fi network as the board.

The Linux OS can be also operated by creating a new terminal in Jupyter Notebooks, which allows root access to the OS.

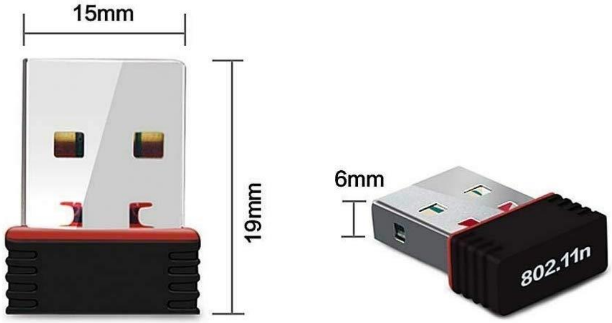


Figure 23 Ralink 5370 Wi-Fi USB Dongle

The DVS video feed can be transmitted to the processing server by mapping the PYNQ board as a network drive and access it from the hub. Other methods that are more robust are currently investigated. Potentially a bash script can be created that automatically sends the data, instead of having to rely on the network drive method. A bash script is a plain text file that contains instructions for the Linux OS.

3.4.4 Passive Infrared Motion Sensor

As mentioned in section 3.4.2 Software Defined Radio, the DVS monitoring system requires a “trigger” that starts transmitting the DVS feed to the processing system. However, since there is no longer an SDR receiver that can act as a “radar” another approach had to be taken. The most evident solution was to use a motion detection sensor for this task, which is the Pmod Passive Infrared (PIR) motion sensor sold by Digilent.

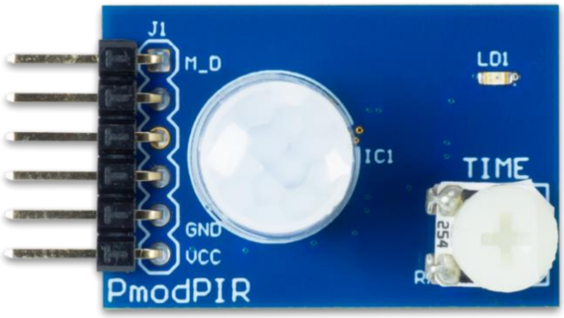


Figure 24 Passive Infrared Motion Sensor Pmod

As seen in Figure 24, the PIR sensor uses the following pins: Vcc, GND and Out. The Output pin goes High when movement is detected. The other pins are not connected. The potentiometer labelled "TIME" changes how long the Output pin should stay High when there is motion detected.

This PIR can be connected to either Pmod A or B on the PYNQ-Z2 board and used through the pre-existing PYNQ PIR motion sensor module [75]. The hardware tests showed that the SoC board was able to successfully detect movement by using this PIR sensor and therefore it can be used to "trigger" the DVS video feed broadcast.

3.5 Fall Classification Using a Neural Network

3.5.1 Data Pre-processing Script

To train a neural network to detect falls from DVS data, a large amount of DVS data is required. Since there are no existing datasets containing DVS data, the dataset had to be created. This meant that video from a conventional camera had to be sourced, and then processed so that it resembled the data from a DVS sensor. To do this, a script had to be written that would convert a video file to a series of DVS images. It is important to note that this code is written purely for processing the dataset for the neural network, and will not be deployed on the final product.

Before the code was written, the script was designed using pseudocode, which can be seen below in Figure 25:

```
load video to be processed
split video into single frames
convert each frame to greyscale
for each frame
    find absolute difference in frame compared to previous frame
    threshold each pixel

save images
```

Figure 25: Pseudocode for Pre-processing Script

This task was initially attempted in MATLAB, though some issues were encountered: while MATLAB was able to easily subtract one frame from the other and provide thresholding, there was no functionality for splitting a video into frames. As a result, the decision was made to re-write the script using python.

The script was implemented using python, and the video file used for testing was split into individual frames using the OpenCV python library [76]. However, finding the difference between the two frames was significantly more difficult in python compared to MATLAB. Significant effort was made to try and produce accurate DVS data in python, but accurate results could not be produced. Due to the time constraints of the project, it was decided that python and MATLAB code should be combined, so that full functionality could be achieved.

A MATLAB script was written that first called a python script, that split a video into individual frames, and then the DVS processing was performed using MATLAB code. This was implemented successfully, and the script was able to pre-process the data for training a DNN in the future. Examples of images produced by the script for a video of a woman walking can be seen below:

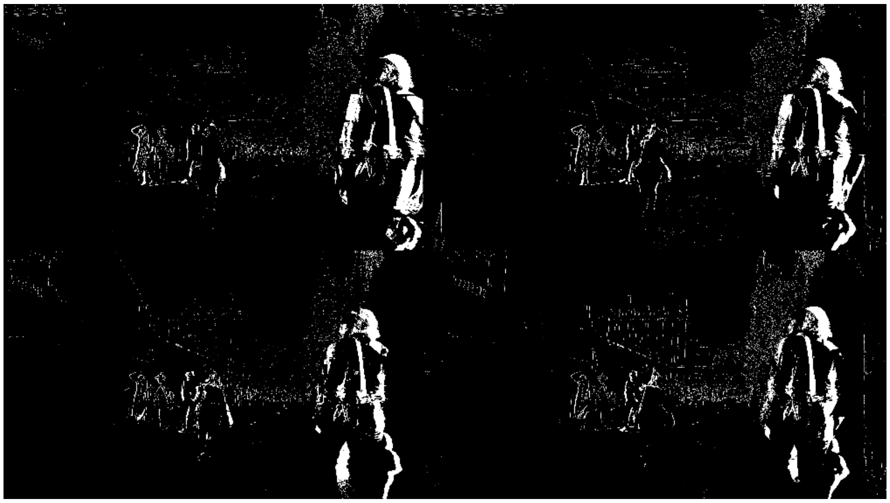


Figure 26: Example DVS Frames from Pre-processing Script

3.6 Voice Recognition

3.6.1 System Overview

Figure 27 shows a high-level diagram of the voice recognition system.

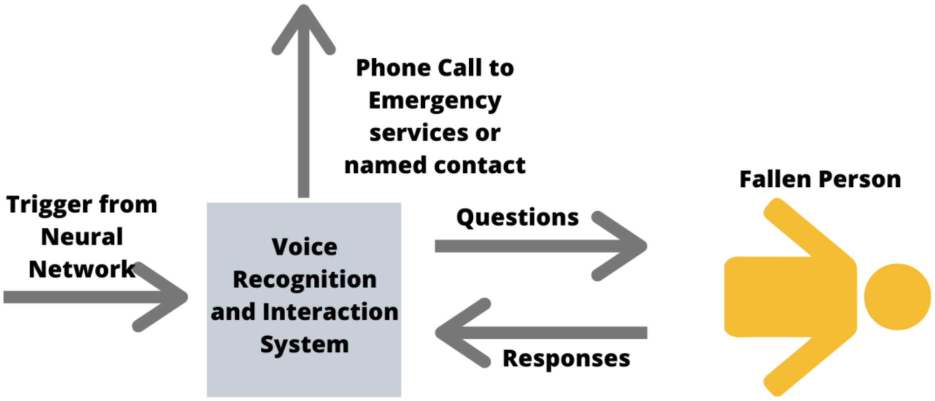

Phone Call to Emergency services or named contact
Trigger from Neural Network
Voice Recognition and Interaction System
Questions
Responses
Fallen Person

Figure 27: Voice Recognition System Design

The voice recognition system is triggered by the fall classification system if it believes there has been a fall. Upon being triggered, the voice recognition system will use text-to-speech to talk to the fallen person and ask them questions to determine what kind of help they require. In order to limit the vocabulary to keep the system simple and accurate, the questions were designed to need only a yes or no response. This has the benefit of being very simple to understand for the less tech-literate users.

Depending on the user's response, the system will either call the emergency services, a named contact, or turn off the system. If the user goes through all of the options and still wants help, the system will ask the questions again. This allows for a wrong choice by the user or wrong recognition on the system's part in an earlier stage. This will only happen a set number of times before it is assumed that something wrong and an ambulance will be called. Figure 28 describes the series of questions and the associated action.

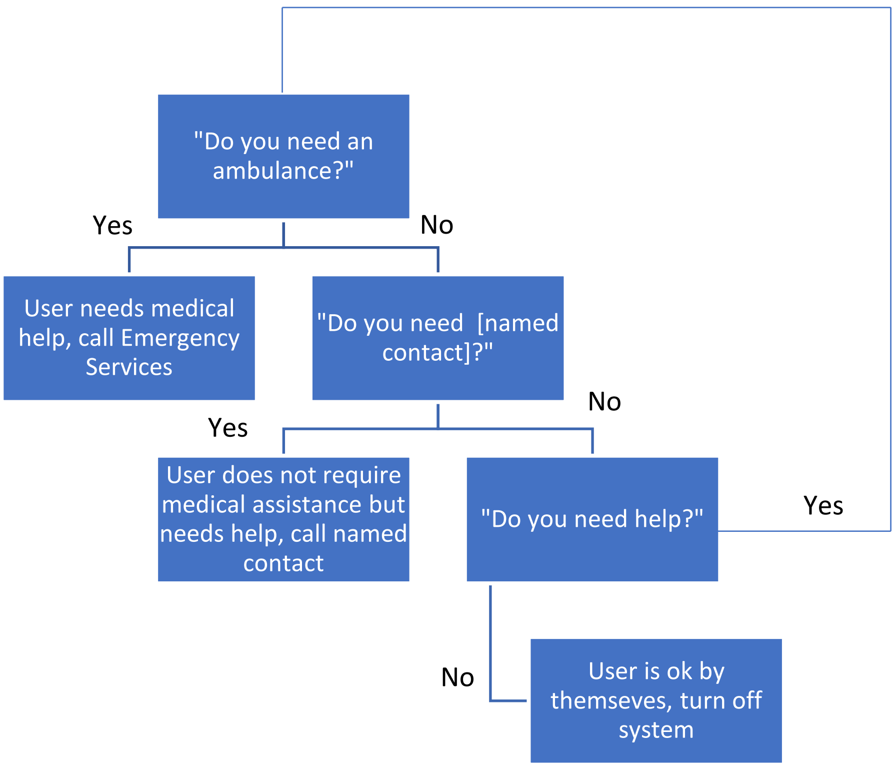

"Do you need an ambulance?"
Yes
No
User needs medical help, call Emergency Services
"Do you need [named contact]?"
Yes
No
User does not require medical assistance but needs help, call named contact
"Do you need help?"
Yes
No
User is ok by themseves, turn off system

Figure 28: Flow Diagram of Questions, Responses, and Actions

Like the fall classification system, it is vital that this section is accurate. False positives are something very undesirable as we do not wish to call the emergency services unduly. Overall, this system aims to reduce the number of calls to the emergency services by offering users the option to call a named contact, however it is important to assume the worst case scenario if there is no response, or an incoherent response is obtained. This is because the user could be unconscious, suffering from something that causes their speech to be unintelligible such as a stroke or high level of pain, or the situation is too noisy for the system to hear. It is vital that the emergency services are called when we are uncertain of the person's condition because it has been shown that the longer an elderly person lies on the ground after falling, the higher the fatality rate gets [77]. Therefore, it is better to have a false positive where the emergency services are called when they are not needed as opposed to a false negative where the fallen person is unconscious and in need of treatment, but no ambulance is called.

3.6.2 Chosen Dataset

The Google Speech Command Dataset is free and contains over 65000 clips of 30 words by 1000 people [78]. This dataset was created to train neural networks, but is ideal for testing this system as it contains thousands of clips of the words "yes" and "no" being said clearly by both men and women across a wide range of ages. This data was used for all tests performed and all results given in the following sections of the report.

3.6.3 Basic Speech Recognition

Python’s Speech Recognition library [79], can be used to connect to one of many different speech APIs including the Google Speech API and Microsoft Bing Speech API. It works in a high-level manner and switching between APIs is very simple [80]. All of the APIs offered, bar one, require an online connection. This does not seem ideal for this project as it introduces delays during processing and there may not be an internet connection available in the real-world application. This leaves only CMUSphinx [81] available to the project.

CMUSphinx is open-source and uses a Hidden Markov Model to recognise speech as described in the background section of the report. By default, CMUSphinx uses a model trained on US English. The Speech Recognition library provides a layer of abstraction from the CMUSphinx code, calling all of the appropriate functions behind the scenes. The result of this is that a basic program to recognise speech from an audio file is only a few lines long.

As an initial test, every yes and no file was run through the recognizer and a simple correct/incorrect result was recorded. The files were clean and no noise was added. The results of this can be seen in Table 3 below.

Table 3: Initial Speech Recognition Results

File Type	Correctly Recognised	Incorrectly Recognised	Can’t recognise
Yes	2076	301	4
No	1744	631	4

These results don’t seem as good as you might expect from a curated dataset and a pre-trained statistical model, so the details were investigated further.

Instead of simply marking the results as correct or incorrect, the incorrect results were displayed. Firstly, the files that the speech recogniser could not transcribe in each case turned out to sound like empty files, so they were removed from the dataset.

Secondly, it was found that while many of the display results indeed, were not yes or no, many of them either contained yes or no (for example, in the no category there were many "no sir" results) or they were an informal version of yes or no such as "yeah" or "nah". The informal terms are correct, and should be marked as such, and while cases like "no sir" are unlikely to come up in the real world, a case where someone says "yes, I do" or "no, I'm fine" should also be marked as correct. In other words, if the phrase contains "yes" or "no" it should also be marked correct. Adjusting the process and rerunning the tests gives the results shown in Table 4.

Table 4: Adjusted speech recognition results

File Type	Correctly Recognised	Incorrectly Recognised
Yes	2173	200
No	1801	570

These results are better, however, there are other parts of the results to address. Examining the results further, it is possible to see that there are some incorrect results that sound similar or the same as "yes" or "no". For example, "yet" and "now" both occur several times. These are notable because they have the same initial sound as "yes" or "no". While these have still been heard incorrectly, for this application, this is actually not an issue. The advantage of having a vocabulary of only "yes" or "no" is that they are very different both in terms of the way they sound and the way the signals look. Figure 29 compares a no signal and a yes signal in the time domain.

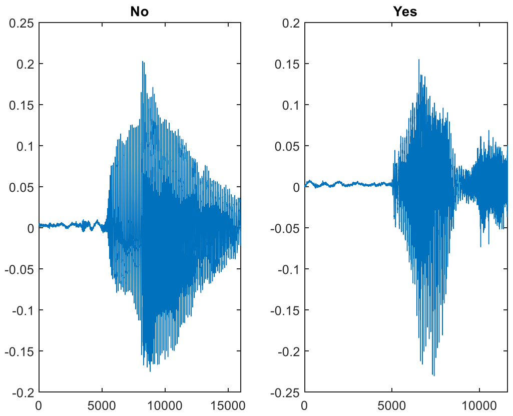


Figure 29: Comparison of No and Yes time domain signal

Therefore, for our application, if we mishear or misclassify “yes” as “yet” we can still be reasonably confident that the user has given a positive response because it is unlikely that “no” would be misheard to contain the “ye” sound.

Additionally, in the case of “no”, its homonym “know” shows up frequently. In the context of our application, we can be certain that this is actually “no”. The final speech recognition results, counting homonyms and words with the same starting sound are shown in Table 5.

Table 5: Final Speech Recognition Results

File Type	Correctly Recognised	Incorrectly Recognised	Percentage Correct
Yes	2215	162	93.34%
No	2150	221	90.07%

These inspire a lot more confidence in the system than before, giving very accurate results that are very close to those achieved by the Google Speech API [54].

3.6.4 Noise Reduction

The results above are for an ideal case where there is no noise. However, there is likely to be at least a small amount of background noise in a normal household and the system must be able to deal with that. 7 common household noises (washing machine, kettle, cat, dog, phone, music, and microwave) were obtained from a stock audio website [82], and the above results were rerun for a single yes and single no signal with each noise signal. Table 6 and Table 7 detail how both humans and the hidden Markov model handle the addition of noise.

Table 6: Results for yes signal with Noise

Noise Type	Recognisable by Humans?	Voice Recogniser output
Kettle	Yes	Yes
Washing machine	Yes	Yes

Phone	Yes	Oh oh
Cat	Yes	Wham
Dog	Yes	[silence]
TV Babble	Yes	Yes sir
Music	Yes	Yes

Table 7: Results for no signal with noise

Noise Type	**Recognisable by Humans?**	**Voice Recogniser output**
Kettle	Yes	No
Washing machine	Yes	No
Phone	Yes	You
Cat	Yes	Hey
Dog	Yes	Who
TV Babble	Yes	No
Music	Yes	No

As can be seen, the recogniser can handle background noise that is distinct from human speech, such as the washing machine, or random TV babble, however, it struggles with things in the same frequency range as human speech, such as the cat noise. Figure 30 shows how the cat noise many large components in the same frequency region as the speech whereas the washing machine has only a few small components.

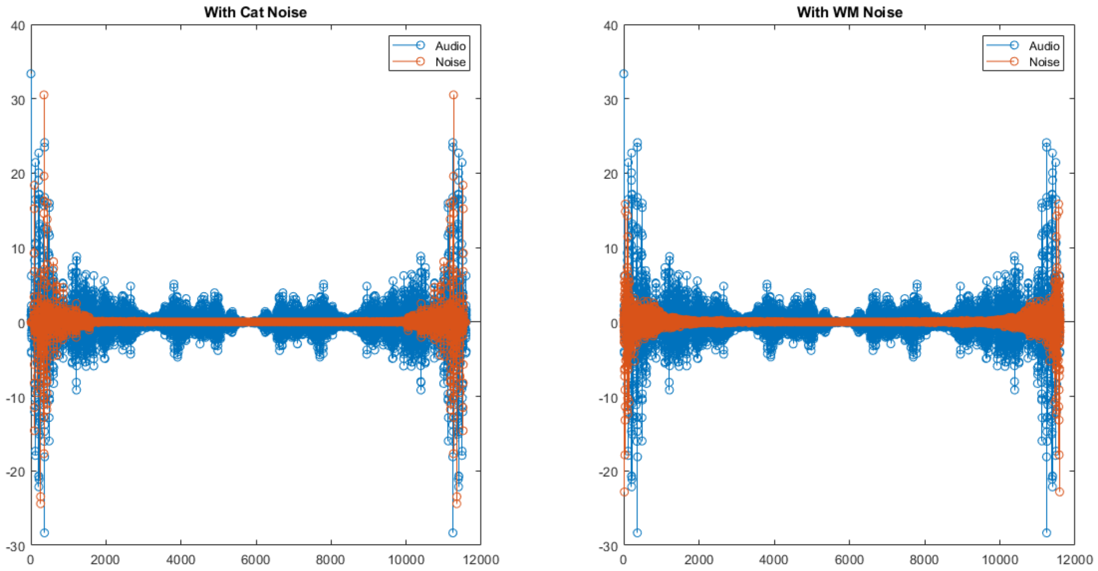


Figure 30: Cat vs. Washing Machine Noise

As mentioned in the background section, there is only so much that can be done to remove noise from a signal without distorting it, however, there are steps can be taken.

First of all, the audio can be bandpass filtered to get rid of anything that isn't within the range of human speech. While this will not help the situation with cat noise, it will get rid of anything that we know is definitely not speech. The frequency range used by telephones is 300-3400 Hz and is designed to contain enough of the harmonic frequencies of human speech to represent the audio faithfully [83], therefore this is the range that the signal will be limited to. This is implemented simply in Python using a low pass and then a high pass filter. As expected, this does not change the results, but it is a step worth keeping.

The desired solution would be able to ignore or remove the noise in a way that causes minimum distortion to the signal. One such method is to estimate the contribution of the noise and subtract it from the spectrum. Estimating the noise may seem a difficult task but because the application is simple and the vocabulary is small, we can use simple pattern matching against a known template, i.e. a clean "yes" or "no" signal. In this case, clean, clear signals were selected to be the templates. Experiments were performed to check whether the gender of the templates mattered and it was found that it did not.

First, FFTs were taken of the template and the noisy signal. Then, the template was subtracted from the noisy signal, giving an approximation of the noise. This approximation can then be subtracted from the noisy signal to obtain a “clean” signal. This process can be seen in Figure 31 where a yes signal has had cat noise added and then removed again.

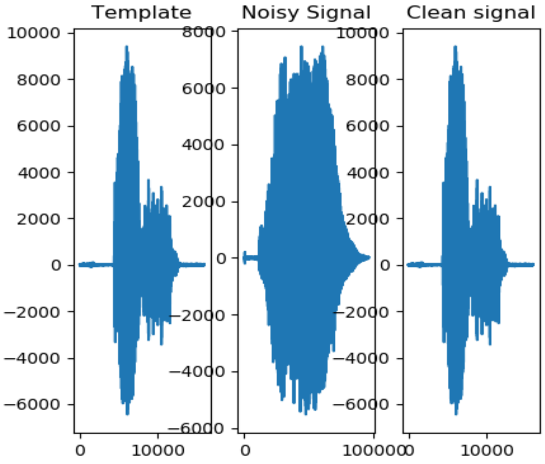


Figure 31: Noise Subtraction Technique (plots in time domain)

This technique works well most of the time as it shapes the noisy signal to be similar to the template. This is good for situations where we know which template to use, however, in practice, we will not know what response to expect. In the worst-case scenario where the noise obscures the signal a lot and the wrong template is used, the signal can actually become a “no” when it should be a “yes” or vice versa. This is the case for the example in Figure 31 which, if a “no” template is used instead, can also be incorrectly turned into a no as shown in Figure 32.

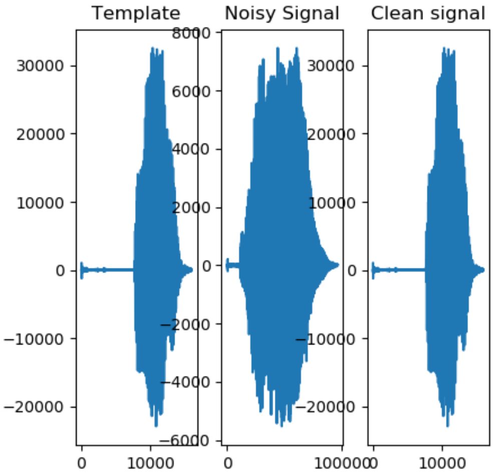


Figure 32: Yes signal incorrectly shaped as a no

This means we must be certain which template to use before implementing this technique.

This can be done by looking at the cross-correlation of each template with the signal. The correlation will be higher when the noisy signal is more similar to the template. In theory, even a noisy signal should be more similar to the correct template than the opposite word. However, in practice, this suffers the same issue as the template matching: if the signal is too obscured it is very easily mistaken for the wrong answer. With the problem test cases such as the cat noise, this is a very unreliable method. Therefore, other options must be explored.

Adaptive noise filters were also investigated. These filters take a desired signal and a realistic signal (i.e. desired signal with noise) and change the weights of the filters so that the difference between the desired signal and real signal is minimised [84]. They are good with low frequency background noise, i.e. noise that is always present (an example in this case might be a washing machine), as it can learn how to get rid of it, however, it is bad with high frequency noises, i.e. noises that are short and loud, like a dog barking. This suggests it might not solve all of the issues in this section either, as the noise that the system struggles with are generally short, thus making them hard to adapt to.

Additionally, for a noise reduction case the filter would need a microphone getting noise and speech and a microphone getting only noise. This is difficult in this application because even if two microphones were used they would need to be in the same room and there would be no way of guaranteeing that one would only pick up noise. This is a major issue because if the second microphone picks up speech as well then the filter will adapt to filter it out too. Therefore, adaptive filters seem unsuitable for this case.

Another possible solution is to simply record the background noise of the room for a few seconds before asking the user to speak to get an approximation of the noise in the room, and then using this to remove noise from the spoken signal, however, this needs more investigation and testing.

3.6.5 Text To Speech

gTTS is a Python library that connects to a Google Text-to-Speech API to return an MP3 of text-to-speech [85]. As mentioned previously, preferably the project would not connect to an API where possible to reduce the delay in the system, however, gTTS can be used to save the mp3 to load in later, offline. Given that the system only requires a few sentences and the fact that these sentences are unlikely to change through the course of the project, this is the method that is used in the project.

During implementation and testing, it was discovered that gTTS won't pronounce names e.g. Andrew. Therefore, the second question in the system will have to refer to "your named contact" or "your selected contact" instead.

3.6.6 Using a Microphone

As mentioned in the background section of the report, the best way to remove noise from the signal is to have a high-quality microphone set-up that doesn't pick up the noise to begin with [59]. One way to do this is to set the microphone sensitivity level lower, so it picks up less background noise when recording, however, the result of this is a quiet speech signal that can be hard to classify. Therefore, pre-amps are often used to "boost" the power of the signal and clean it a little to improve the quality [86].

There are many high-quality microphones with good pre-amps built-in available, however, most of these are designed for studio applications such as recording music or voice for things like audiobooks, television shows, and music albums. While one of these could be used for our project, ideally the microphone would be small and unnoticeable to help alleviate the feeling of being listened to on the user.

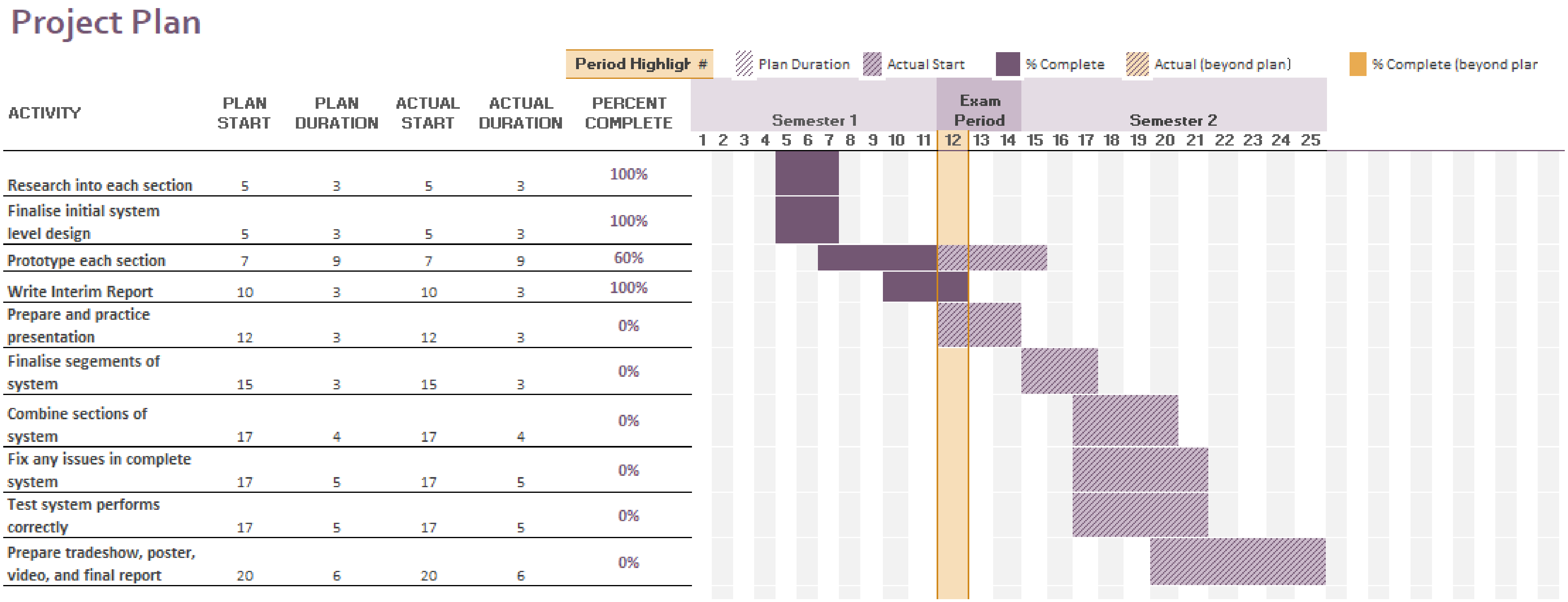
A more suitable microphone was found in the form of a microphone for home security. It is small, designed to be stuck to a wall or ceiling and be unnoticeable, and has a preamp built-in so that it can have the microphone at a very low level and still record speech from up to 30 ft away. It comes with a long cable to allow it to be positioned easily and it is also relatively low-cost. This makes it an ideal choice for the project.

Recording from a microphone is done simply in Python using the PyAudio library [87].

4 Discussion

4.1 General Discussion

The project as a whole is moving forward as expected. All hardware that the system is expected to require has been purchased, and the objectives for the end of the first semester have been met. The cost of the project is well within the budget, and the bill of expenses can be seen in appendix 7.1. As well as meeting the technical objectives, the group has worked well as a team: the group has met on a weekly basis; and regular communication the group mentor and proposer has been maintained. The updated Gantt chart with the current progress can be seen below in Figure 33.



Project Plan

ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Research into each section	5	3	5	3	100%
Finalise initial system level design	5	3	5	3	100%
Prototype each section	7	9	7	9	60%
Write Interim Report	10	3	10	3	100%
Prepare and practice presentation	12	3	12	3	0%
Finalise segements of system	15	3	15	3	0%
Combine sections of system	17	4	17	4	0%
Fix any issues in complete system	17	5	17	5	0%
Test system performs correctly	17	5	17	5	0%
Prepare tradeshow, poster, video, and final report	20	6	20	6	0%

Figure 33: Gantt Chart with Current Progress

4.2 Camera Interface

A great deal of research has been done into methods and products for receiving image data on the PYNQ Board. Various interfaces were explored along with products available to analyse each of them for viability as a solution for the project. Each was examined on their ease of implementation with the PYNQ Board, but also in terms of performance and data rates. A High FPS would be desired in the camera to allow for a better performance of the whole system, so this was taken largely into account.

CSI-2 was chosen as the preferred method camera interface as there are many available at different price points. The cheaper cameras boast a configurable 90+ fps which is more than other methods, and if this is successfully implemented it is easy to improve the performance of the system just by replacing the camera with a more expensive one. Both a camera and adapter has been sourced for prototyping the use of this interface.

However, due to the complexity of implementing this interface with the IP Block, the next steps in the project will be focused on a preliminary prototype. A HDMI or USB Camera will be used that is compatible with the PYNQ Board to obtain images and process them into DVS Data and create a fully working basic system. The interface with the CSI-2 Camera will become a priority after a basic system has been designed.

4.3 FPGA Design

The progress up to now has mainly been research and tutorial based. Many new elements have been introduced and investigated, mainly the process of emulating DVS events as well as the differences in PYNQ development compared to a traditional Vivado IP Integrator design flow.

This has resulted in the evaluation of an IP block which can input and process raw camera data into a useful AXI format alongside the example design packaged alongside this block. The example design is seen to be a strong introduction to the use of the IP block however, this approach did not take advantage of the PYNQ design flow, using an overlay to design the PL and PS aspects of the design together. Therefore, research and tutorials are undertaken to give a more comprehensive understanding of this design methodology. The default base overlay is seen to be powerful, with a lot of in-built functionality for IO such as HDMI, USB, PMOD and several detailed example programs to ease overlay development. The learning approach up to now has been large however results should now begin to be seen shortly.

The progress up to now sets this section of the project up effectively to implement a camera to board connection soon, followed shortly after by a DVS implementation. To do so a simple approach will be taken first, using a HDMI or USB camera, before implementing the chosen MIPI camera. This is a considerably simpler approach as there is support within the pre-installed base overlay for the PYNQ-Z2 for HDMI and USB input. It should be a straightforward case of following the steps of a provided tutorial to receive input frames and output them as a video stream to a HDMI screen. This will provide a foundation to design the emulated DVS events. This simple model can be added to via python without needing to worry about the underlying PL design as this is handled by the base overlay. If this can be shown to be a success then this part of the project will be successful, DVS images from a standard camera. However, this solution will be limited by the camera itself, HDMI and USB cameras are both limited in framerate. While a device may have an internal refresh rate of 120 Hz or 240 Hz, there are no digital devices that will emit a signal at this framerate. There are no "120 Hz" HDMI Cables that exist, as video feeds from devices very rarely are higher than 60 fps [88]. USB2.0 Cameras require drivers that may not all be compatible with the limited OS on the PYNQ board, and provide slow data rates in comparison to other interfaces (Up to 60 MB/s, typically 30 MB/s when compared to CSI-2, which can provide up to 10 GB/s [89] [30].

This can be improved by implementing the method using the chosen MIPI camera connected to the PYNQ-Z2 via PMOD. This approach will require edits to the IP integrator model within the base overlay, incorporating the identified MIPI CSI-2 Receiver Subsystem, a process which will require further research into custom overlays.

4.4 Wireless data transmission and passive infrared sensor module

The progress done on the wireless communications system has exceeded the initial expectations, as at the moment the PYNQ-Z2 can be remotely accessed via a PC or another device that is connected to the same network. The pre-existing PYNQ framework and the base overlay played an important role in this breakthrough, as it was straightforward to use the Wi-Fi dongle since the driver was already created.

As a result, the board can be remotely programmed via the overlay system, available through the PYNQ framework. A simple LED demo can be deployed via Jupyter Notebooks which remotely toggles the LEDs on the PYNQ board.

The Pmod PIR sensor was also successfully connected and tested on the PYNQ-Z2. Another demo was created which toggles the LEDs on the PYNQ board when the sensor detects movement around it. The only physical connection needed by the board is to an external power source to turn on the board.

4.5 Fall Classification Using a Neural Network

The pre-processing script has been written and tested, and this will be used to modify the dataset for training the DNN. In the next semester, an image classification network will be selected, trained, and tested to ensure that the system is able to accurately determine whether the user has fallen. The fall classification system will then be combined with the rest of the project, and tested with data from the low-cost DVS sensor.

4.6 Voice Recognition System

The system can successfully communicate with a person, recognise them speaking, and use the answers to make a decision. Several noise reduction techniques have been investigated and tested and a method has been chosen. The remaining steps for the voice recognition system is further research and testing into noise reduction, implementation of the section to make phone calls to the appropriate person, and tests of the whole voice recognition system. If there is time, an option to recognise further words such as "stop" may be implemented to catch cases where an ambulance is called mistakenly. Additionally, a universal infrared controller could be used to turn off a television or similar electronic in order to reduce the noise in the room if that remains a problem.

5 Conclusion

In summary, the project has progressed as expected, and is on track to be completed on time. This has been achieved by making a high-level system design that meets the design criteria, researching relevant background information, and making progress on implementing the components of the project. In semester two, it is expected that the individual parts of the system will be completed, and will then be combined to create a working finished product. The project will then be demonstrated at the tradeshow.

No significant issues have been encountered so far, and the possible risks outlined at the start of the project will be avoided through regular meetings and continuous work on the project. If problems are encountered, then the mitigating action outlined in Table 2 will be taken to reduce the impact.

6 References

[1] World Health Organisation, “Falls,” 16 January 2018. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/falls. [Accessed 23 November 2019].

[2] NHS, “Falls Overview,” [Online]. Available: https://www.nhs.uk/conditions/falls/. [Accessed 23 November 2019].

[3] D. Wild, U. S. Nayak and B. Isaccs, “How dangerous are falls in old people at home?,” *British Medical Journal,* vol. 282, pp. 266-268, 1981.

[4] K. Anderson, “Falls in the elderly,” Royal Collage of Physicians of Edinburgh, Edinburgh, 2008.

[5] A. O. S. R. A. P. R. M. A. S. P. M. J. d. A. N. L. F. R. T. Adriana de Azevedo Smith, “Assessment of risk of falls in elderly living at home,” 6 April 2017. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5396481/. [Accessed 23 November 2019].

[6] “Homepage,” Lifeline24, [Online]. Available: https://www.lifeline24.co.uk/. [Accessed 23 November 2019].

[7] M. S. M. S. B. K. K. S. L. M. P. a. F. I. Mohammad Ashfak Habib, “Smartphone-Based Solutions for Fall Detection and Prevention: Challenges and Open Issues,” 14 April 2014. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4029687/. [Accessed 23 November 2019].

[8] M. G. C. a. A. J. G. Eleftheria Vaportzis, “Older Adults Perceptions of Technology and Barriers to Interacting with Tablet Computers: A Focus Group Study,” 4 October 2017. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5649151/. [Accessed 23 November 2019].

[9] Inivation, “The Dynamic Vision Sensor,” Inivation, 2019. [Online]. Available: https://inivation.com/dvs/. [Accessed 25 11 2019].

[10] P. Lichtsteiner, C. Posch and T. Delb, “A 128× 128 120 dB 15 μs Latency Asynchronous Temporal Contrast Vision Sensor,” *IEEE Journal of Solid-State Circuits,* vol. 43, no. 2, pp. 566 - 576, 2008.

[11] Samsung, “SmartThings Vision,” Samsung, 2019. [Online]. Available: https://www.samsung.com/au/mobile-iot/smartthings-vision-u999/. [Accessed 25 11 2019].

[12] N. Owano, “Dynamic Vision Sensor tech works like human retina,” Phys.org, 2013. [Online]. Available: https://phys.org/news/2013-08-dynamic-vision-sensor-tech-human.html. [Accessed 05 12 2019].

[13] G. P. Garcia, P. Camilleri, Q. Liu and S. Furbur, “pyDVS: An Extensible, Real-time Dynamic Vision Sensor Emulator using Off-the-Shelf Hardware,” IEEE, 2016.

[14] L. Crockett, *Introduction to FPGAs for DSP - EE580 Lecture Notes,* University of Strathclyde, 2019.

[15] BusinessWire, “Global Field Programmable Gate Array (FPGA) Market Outlook to 2024 - SRAM-based FPGAs to Account for a Significant Share - ResearchAndMarkets.com,” BuinessWire, 29 7 2019. [Online]. Available: https://www.businesswire.com/news/home/20190729005380/en/Global-Field-Programmable-Gate-Array-FPGA-Market. [Accessed 27 11 2019].

[16] Xilinx, “ZynqUltraScale+ RFSoCData Sheet: Overview,” Xilinx, 27 August 2019. [Online]. Available: https://www.xilinx.com/support/documentation/data_sheets/ds889-zynq-usp-rfsoc-overview.pdf. [Accessed 02 12 2019].

[17] Xilinx, “Moving a Generation Ahead with All Programmable FPGAs, SoCs, and 3D ICs,” 2012. [Online]. Available: https://www.xilinx.com/publications/prod_mktg/Generation-Ahead-Backgrounder.pdf. [Accessed 30 11 2019].

[18] F. Fallahlalehzari, “Introduction to Zynq™ Architecture - A brief examination of the Zynq processing system and its programmable logic,” Aldec - The Design Verification Company, 2019. [Online]. Available: https://www.aldec.com/en/company/blog/144--introduction-to-zynq-architecture. [Accessed 27 11 2019].

[19] L. Crockett, *FPGA and SoC Architecture Fundamentals - EE580 Lecture Notes,* University of Strathclyde, 2019.

[20] Xilinx, “Zynq-7000 SoC,” Xilinx, [Online]. Available: https://www.xilinx.com/products/silicon-

devices/soc/zynq-7000.html#productAdvantages. [Accessed 02 12 2019].

[21] Xilinx, "Zynq-7000 product brief," Xilinx, 2018. [Online]. Available: https://www.xilinx.com/support/documentation/product-briefs/zynq-7000-product-brief.pdf. [Accessed 02 12 2019].

[22] Xilinx, "PYNQ: PYTHON PRODUCTIVITY FOR ZYNQ," Xilinx, [Online]. Available: http://www.pynq.io/. [Accessed 30 11 2019].

[23] Xilinx, "PYNQ_Workshop/Session_1/4_Programming_onboard_peripherals.ipynb," Xilinx, 17 1 2019. [Online]. Available: https://github.com/Xilinx/PYNQ_Workshop/blob/master/Session_1/4_Programming_onboard_peripherals.ipynb. [Accessed 05 12 2019].

[24] Xilinx, "Video using the Base Overlay," Xilinx, 2016. [Online]. Available: https://pynq.readthedocs.io/en/v1.3/9b_base_overlay_video.html. [Accessed 05 12 2019].

[25] Xilinx, "Vivado Design Suite HLx Editions - Accelerating High Level Design," Xilinx, 2019. [Online]. Available: https://www.xilinx.com/products/design-tools/vivado.html. [Accessed 05 12 2019].

[26] Xilinx, "Accelerating Integration - Block-based IP Integration with Vivado IP Integrator," Xilinx, 2019. [Online]. Available: https://www.xilinx.com/products/design-tools/vivado/integration.html. [Accessed 05 12 2019].

[27] Topituuk, "Wikipedia - Software Defined Radio Scheme, Copyrighted free use," 25 12 2009. [Online]. Available: https://commons.wikimedia.org/w/index.php?curid=8831874. [Accessed 02 12 2019].

[28] R. B. K. A. D. a. C. L. Stewart, Software defined radio using MATLAB & Simulink and the RTL-SDR, Glasgow: University of Strathclyde, 2015.

[29] T. V. W. &. B. J. Marshall Brain, "Howstuffworks," [Online]. Available: https://computer.howstuffworks.com/wireless-network1.htm. [Accessed 04 12 2019].

[30] MIPI, "MIPI Overview," MIPI Alliance, [Online]. Available: https://www.mipi.org/about-us.

[31] Mouser, “Camera module Cameras & Camera Modules,” Mouser, [Online]. Available: https://www.mouser.co.uk/Optoelectronics/Cameras-Accessories/Cameras-Camera-Modules/_/N-fb8v9?Keyword=camera+module&FS=True.

[32] FTDI, “What is the Camera Parallel Interface?,” 23 03 2015. [Online]. Available: https://www.ftdichip.com/Support/Documents/TechnicalNotes/TN_158_What_Is_The_Camera_Parallel_Interface.pdf.

[33] A. Williams, “MIPI CSI-2 Implementation in FPGAS,” 29 11 2018. [Online]. Available: https://hackaday.com/2018/11/29/mipi-csi-2-implementation-in-fpgas/.

[34] A. Taylor, “MicroZed Chronicles: Working with MIPI,” [Online]. Available: https://www.hackster.io/news/microzed-chronicles-working-with-mipi-b16b67990ccc.

[35] M. Alliance, “MIPI Display Serial Interface (MIPI DSI),” MIPI Alliance, [Online]. Available: https://www.mipi.org/specifications/dsi.

[36] D. Faggella, “What is Machine Learning?,” Emerj, 21 November 2019. [Online]. Available: https://emerj.com/ai-glossary-terms/what-is-machine-learning/. [Accessed 22 November 2019].

[37] K. Didur, “Machine learning in finance: Why, what & how,” Towards Data Science, 11 July 2018. [Online]. Available: https://towardsdatascience.com/machine-learning-in-finance-why-what-how-d524a2357b56. [Accessed 22 November 2019].

[38] J. S. B. C. M. A. O. A. E. A. Emmanuel Gbenga Dada, “Machine learning for email spam filtering: review, approaches and open research problems,” 10 June 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405844018353404. [Accessed 22 November 2019].

[39] I. Baikalov, “Applications and limitations of machine learning in cybersecurity,” ITProPortal, 05 July 2019. [Online]. Available: https://www.itproportal.com/features/applications-and-limitations-of-machine-learning-in-cybersecurity/. [Accessed 22 November 2019].

[40] “Everything You Need to Know About Artificial Neural Networks,” Medium, 28 December 2015. [Online]. Available: https://medium.com/technology-invention-and-more/everything-you-need-to-

know-about-artificial-neural-networks-57fac18245a1. [Accessed 01 December 2019].

[41] J. Dacombe, “An introduction to Artificial Neural Networks,” 23 October 2017. [Online]. Available: https://medium.com/@jamesdacombe/an-introduction-to-artificial-neural-networks-with-example-ad459bb6941b. [Accessed 06 December 2019].

[42] B. Marr, “Deep Learning Vs Neural Networks - What’s The Difference?,” Bernard Marr & Co, [Online]. Available: https://bernardmarr.com/default.asp?contentID=1789. [Accessed 06 December 2019].

[43] S. Saha, “A Comprehensive Guide to Convolutional Neural Networks,” 15 December 2018. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. [Accessed 06 December 2019].

[44] V. Bushaev, “How do we ‘train’ neural networks ?,” Towards Data Science, 27 November 2017. [Online]. Available: https://towardsdatascience.com/how-do-we-train-neural-networks-edd985562b73. [Accessed 2019 December 2019].

[45] M. Rouse, “Speech Recognition,” TechTarget, 06 12 16. [Online]. Available: https://www.google.co.uk/amps/s/searchcustomerexperience.techtarget.com/definition/speech-recognition%3famp=1. [Accessed 21 11 19].

[46] “Apple Voice Recognition Overview,” LearningApps, [Online]. Available: https://www.learningapps.co/uk/moodle/xertetoolkits/play.php?templated_id=1315. [Accessed 21 11 19].

[47] LearningApps, “Voice Recognition Android,” [Online]. Available: https://www.learningapps.co.uk/moodle/xertetoolkits/play.php?template_id=1278. [Accessed 21 11 19].

[48] Five9, “How IVR and Voice Recognition Software Can Help A Small Business,” [Online]. Available: https://www.five9.com/delight-customers-with-ivr-done-right. [Accessed 21 11 19].

[49] HowStuffWorks, “Speech Recognition,” [Online]. Available: electronics.howstuffworks.com/gadgets/high-tech-gadgets/speech-recognition.htm. [Accessed 21

11 19].

[50] devteam, “How to make a speech recognition system,” [Online]. Available: devteam.space/blog/hot-to-make-a-speech-recognition-system/. [Accessed 21 11 19].

[51] L. Cuiling, “English Speech Recognition Method Based on Hidden Markov Model,” *International Conference on Smart Grid and Electrical Automation,* 2016.

[52] R. S. Chavan and S. S. Ganesh, “An overview of speech recognition using HMM,” *International Journal of Computer Science and Mobile COmputing,* vol. 2, no. 6, pp. 233-238, 2013.

[53] Google, “Google Speech To Text,” [Online]. Available: https://cloud.google.com/speech-to-text/. [Accessed 21 11 19].

[54] A. Li, “Google's speech recognition is now almost as accurate as humans,” 1 6 17. [Online]. Available: www.google.co.uk/amp/s/9to5google.com/2017/06/01/google-speech-recognition-humans/amp/. [Accessed 21 11 19].

[55] Quora Users, “Which is better for a speech recognition system? The neural network or the hidden markov model,” [Online]. Available: https://www.quora.com/Which-is-better-in-terms-of-accuracy-for-a-speech-recognition-system-the-Neural-Network-or-the-Hidden-Markov-Model. [Accessed 21 11 19].

[56] H. v. Lier, “A Basic Introduction to Speech Recognition (Hidden Markov Model & Neural Networks),” 2 9 18. [Online]. Available: youtube.com/watch?v=U0xtE4_QLXI. [Accessed 22 11 19].

[57] WikiBooks, “Welsh/Alphabet,” [Online]. Available: https://en.wikibooks.org/wiki/Welsh/Alphabet. [Accessed 22 11 19].

[58] CMU Sphinx, “CMU Sphinx Open Source Models,” [Online]. Available: http://www.speech.cs.cmu.edu/sphinx/models/. [Accessed 22 11 19].

[59] B. Desthuilliers, “How to ignore background noise while recording audio using pyaudio in python?,” [Online]. Available: https://stackoverflow.com/questions/54090371/how-to-ignore-background-noise-while-recording-audio-using-pyaudio-in-python. [Accessed 22 11 19].

[60] A. Williams, “MIPI CSI-2 Implementation in FPGAs,” 29 11 2018. [Online]. Available: https://hackaday.com/2018/11/29/mipi-csi-2-implementation-in-fpgas/.

[61] "Jippie", “System on Chip - Camera Parallel Interface vs Camera Serial Interface,” Electronics Stack Exchange, [Online]. Available: https://electronics.stackexchange.com/questions/153563/camera-parallel-interface-vs-camera-serial-interface.

[62] OpenCV, “OpenCV-Python Tutorials,” OpenCV, 2019. [Online]. Available: https://docs.opencv.org/master/d6/d00/tutorial_py_root.html. [Accessed 30 11 2019].

[63] Technopedia, “Why is python so popular in machine learning?,” [Online]. Available: https://www.techopedia.com/why-is-python-so-popular-in-machine-learning/7/32881. [Accessed 29 11 19].

[64] Xilinx, “MIPI CSI-2 Receiver Subsystem v4.1 Product Guide,” 22 11 2019. [Online]. Available: https://www.xilinx.com/support/documentation/ip_documentation/mipi_csi2_rx_subsystem/v4_1/pg232-mipi-csi2-rx.pdf. [Accessed 1 12 2019].

[65] Xilinx, “PYNQ Workshop,” Xilinx, 3 2019. [Online]. Available: https://github.com/Xilinx/PYNQ_Workshop. [Accessed 05 12 2019].

[66] Xilinx, “PYNQ Workshop Session 1,” Xilinx, 1 2019. [Online]. Available: https://github.com/Xilinx/PYNQ_Workshop/tree/master/Session_1. [Accessed 05 12 2019].

[67] Xilinx, “PYNQ Workshop Session 2 - Introduction to Overlays,” Xilinx, 26 11 2018. [Online]. Available: https://github.com/Xilinx/PYNQ_Workshop/tree/master/Session_2. [Accessed 05 12 2019].

[68] Xilinx, “USB Webcam,” Xilinx, 21 3 2018. [Online]. Available: https://github.com/Xilinx/PYNQ/blob/master/pynq/notebooks/common/usb_webcam.ipynb. [Accessed 05 12 2019].

[69] P. S. P. S. N. J. I. B. K. G. Kate T., “"Bluetooth vs Wi-Fi." Diffen.com,” [Online]. Available: https://www.diffen.com/difference/Bluetooth_vs_Wifi. [Accessed 05 12 2019].

[70] Digilent, “Pmod AD5: 4-channel 4.8 kHz 24-bit A/D Converter,” Digilent, [Online]. Available:

https://store.digilentinc.com/pmod-ad5-4-channel-4-8-khz-24-bit-a-d-converter/. [Accessed 06 12 2019].

[71] Digilent, “Pmod DA3: One 16-bit D/A Output,” Digilent, [Online]. Available: https://store.digilentinc.com/pmod-da3-one-16-bit-d-a-output/. [Accessed 06 12 2019].

[72] Analog Devices, “ADALM-Pluto Product Details,” Analog Devices, [Online]. Available: https://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/ADALM-PLUTO.html#eb-overview. [Accessed 05 12 2019].

[73] "jpeyron", “Digilent Forums,” 17 06 2019. [Online]. Available: https://forum.digilentinc.com/topic/17224-pmod-wifi-sdk-problem/#comment-47597. [Accessed 06 12 2019].

[74] Xilinx, “PYNQ Documentation,” [Online]. Available: https://pynq.readthedocs.io/en/v2.5/pynq_package/pynq.lib/pynq.lib.wifi.html. [Accessed 05 12 2019].

[75] Xilinx, “PYNQ Documentation PIR Sensor,” [Online]. Available: https://pynq.readthedocs.io/en/v2.5/pynq_package/pynq.lib/pynq.lib.pmod.html#module-pynq.lib.pmod.pmod_grove_pir. [Accessed 05 12 2019].

[76] “OpenCV-Python Tutorials,” [Online]. Available: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_tutorials.html. [Accessed 07 December 2019].

[77] D. Wild, N. U. S. and B. Isaccs, “How dangerous are falls in old people at home,” *British Medical Journal,* vol. 282, pp. 266-268, 1981.

[78] P. Warden, “Launching the Speech Commands Dataset,” 24 8 17. [Online]. Available: https://ai.googleblog.com/2017/08/launching-speech-commands-dataset.html. [Accessed 22 11 19].

[79] CMU Sphinx, “SpeechRecognition,” [Online]. Available: pypi.org/project/SpeechRecognition/. [Accessed 22 11 19].

[80] D. Amos, “The Ultimate Guide To Speech Recognition With Python,” RealPython, [Online].

Available: realpython.com/python-speech-recogntion/. [Accessed 22 11 19].

[81] CMUSphinx, “CMUSphinx,” [Online]. Available: https://cmusphinx.github.io/. [Accessed 22 11 19].

[82] ZapSplat, “Free sound effects and royalty free music,” [Online]. Available: zapsplat.com. [Accessed 12 11 19].

[83] Wikipedia authors, “Voice Frequency,” [Online]. Available: en.wikipedia.org/wiki/Voice_frequency. [Accessed 12 11 19].

[84] S. Weiss, “Adaptive Signal Processing,” in *Advanced DSP Lecture Notes*, 2019, pp. 274-290.

[85] PyPi, “gTTS,” [Online]. Available: https://pypi.org/project/gTTS/. [Accessed 4 12 19].

[86] KKSound, “What does a preamp do and do I need one?,” [Online]. Available: https://kksound.com/support/pickups101/preamp.php. [Accessed 29 11 19].

[87] PyAudio, “PyAudio,” [Online]. Available: https://pypi.org/project/PyAudio/. [Accessed 29 11 19].

[88] K. Denke, “HDMI Cable Speed & Features Explained | Audioholics,” Audioholics, 10 March 2010. [Online]. Available: https://www.audioholics.com/audio-video-cables/hdmi-cable-speed.

[89] L. Spector, “USB3.0 Speed: Real and imagined,” PC World, 26 6 2014. [Online]. Available: https://www.pcworld.com/article/2360306/usb-3-0-speed-real-and-imagined.html.

[90] G. Fitzpatrick, K. Purden, A. Burr, L. Brown and D. Nadejde, “19520 Fall Detection DVS,” [Online]. Available: https://github.com/graemef681/19520-Fall-Detection-DVS. [Accessed 06 12 19].

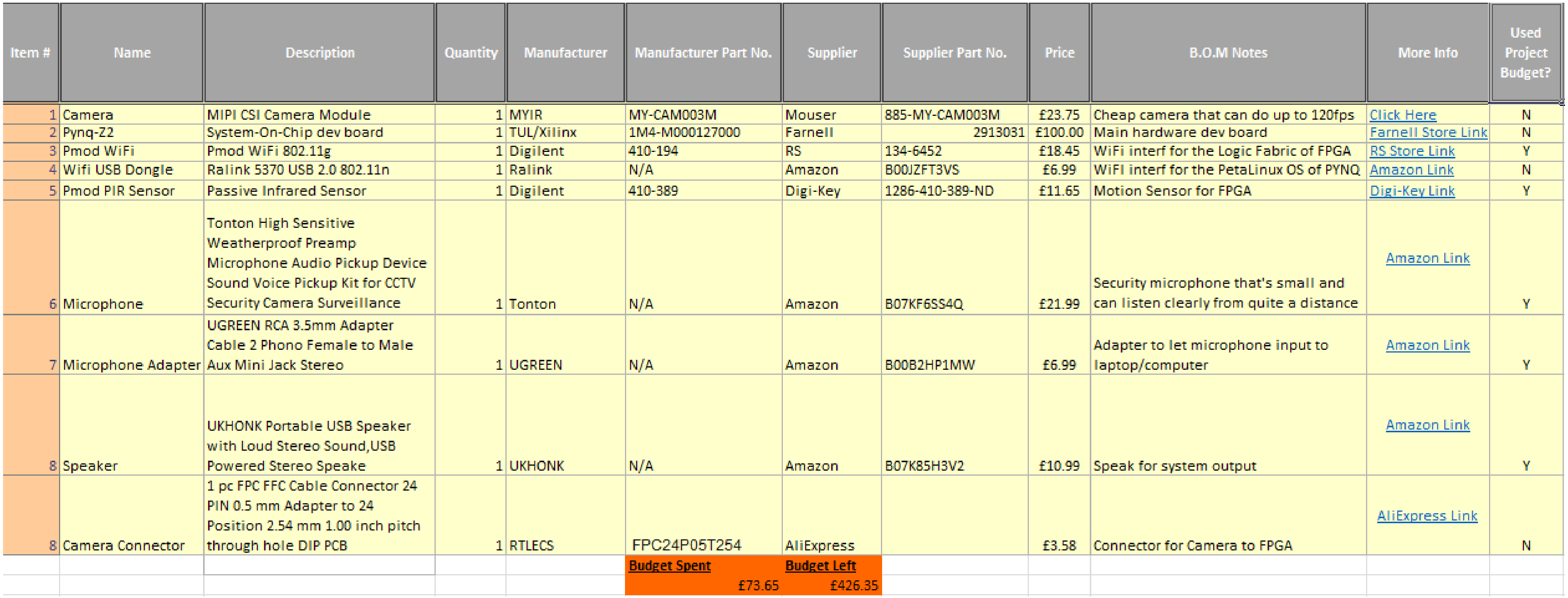
[91] Xilinx, “PYNQ: Introduction to Overlays,” 27 11 2018. [Online]. Available: https://github.com/Xilinx/PYNQ_Workshop/blob/master/Session_2/PYNQ_Workshop_Introduction_To_Overlays.pdf. [Accessed 30 11 2019].

[92] Xilinx, “Introduction to Overlays,” Xilinx, 2016. [Online]. Available: https://pynq.readthedocs.io/en/v1.3/6_overlays.html. [Accessed 05 12 2019].

[93] Inivation, “Buy,” Inivation, 2019. [Online]. Available: https://inivation.com/buy/. [Accessed 05 12 2019].

7 Appendix

7.1 Bill of Expenses



Item #	Name	Description	Quantity	Manufacturer	Manufacturer Part No.	Supplier	Supplier Part No.	Price	B.O.M Notes	More Info	Used Project Budget?
1	Camera	MIPI CSI Camera Module	1	MYIR	MY-CAM003M	Mouser	885-MY-CAM003M	£23.75	Cheap camera that can do up to 120fps	Click Here	N
2	Pynq-Z2	System-On-Chip dev board	1	TUL/Xilinx	1M4-M000127000	Farnell	2913031	£100.00	Main hardware dev board	Farnell Store Link	N
3	Pmod WiFi	Pmod WiFi 802.11g	1	Digilent	410-194	RS	134-6452	£18.45	WiFi interf for the Logic Fabric of FPGA	RS Store Link	Y
4	Wifi USB Dongle	Ralink 5370 USB 2.0 802.11n	1	Ralink	N/A	Amazon	B00JZFT3VS	£6.99	WiFI interf for the PetaLinux OS of PYNQ	Amazon Link	N
5	Pmod PIR Sensor	Passive Infrared Sensor	1	Digilent	410-389	Digi-Key	1286-410-389-ND	£11.65	Motion Sensor for FPGA	Digi-Key Link	Y
6	Microphone	Tonton High Sensitive Weatherproof Preamp Microphone Audio Pickup Device Sound Voice Pickup Kit for CCTV Security Camera Surveillance	1	Tonton	N/A	Amazon	B07KF6SS4Q	£21.99	Security microphone that's small and can listen clearly from quite a distance	Amazon Link	Y
7	Microphone Adapter	UGREEN RCA 3.5mm Adapter Cable 2 Phono Female to Male Aux Mini Jack Stereo	1	UGREEN	N/A	Amazon	B00B2HP1MW	£6.99	Adapter to let microphone input to laptop/computer	Amazon Link	Y
8	Speaker	UKHONK Portable USB Speaker with Loud Stereo Sound,USB Powered Stereo Speake	1	UKHONK	N/A	Amazon	B07K85H3V2	£10.99	Speak for system output	Amazon Link	Y
8	Camera Connector	1 pc FPC FFC Cable Connector 24 PIN 0.5 mm Adapter to 24 Position 2.54 mm 1.00 inch pitch through hole DIP PCB	1	RTLECS	FPC24P05T254	AliExpress		£3.58	Connector for Camera to FPGA	AliExpress Link	N
					Budget Spent	**Budget Left**					
					£73.65	£426.35					

7.2 Statement of Intent

Department of Electronic & Electrical Engineering

MEng/BEng in EEE/CES

19.520 MEng Project

This project registration form is organised in four parts:

PART 1:STATEMENT OF INTENT

PART 2:PROJECT WORK PLAN

PART 3: RESOURCE REQUIREMENT

PART 4: RISK ASSESSMENT

PART 5:SAFETY DECLARATION

1. All parts of the form must be completed jointly by group members and lodged with R4.01(EEE Resource Centre) by 13.30 on 18th October 2019.

1. Copies of the completed form will be sent to the Project Mentor
2. The group is advised to retain a copy of the completed form for future reference.

Group Names

1) Andrew Burr	2) Lewis Brown	3) Graeme Fitzpatrick
4) Daniel Nadejde	5) Kirsty Purden	**6)**
Project Title: Low cost dynamic vision sensor for fall detection system		

PART 1:STATEMENT OF INTENT

The purpose of this section is: (i) to provide a concise description of the project, and (ii) to state a set of objectives that will provide the guide for assessing the project. Students should note the importance of item (ii), which should be discussed in detail with their project sponsor.

A. Project Description:

The group (in consultation with the Project Sponsor if appropriate) describe the project in up to 1000 words:

Falls at home present a significant risk, especially to elderly people over 75 for whom falling is the most common cause of injury related deaths in the UK [1]. The dangers of falling are amplified if the individual at risk lives alone, as they may not be able get the assistance they need in case of an accident. Current solutions to this problem do exist, but they are not ideal. Simple solutions are available like bracelets or necklaces which can activate an alarm if pressed, but these require that the user is wearing the device at all times, and can cause false alarms. More complex home security systems also exist, but these often require a monthly subscription, and can lead to feeling of loss of privacy. As a result, there is a gap in the market for a low-cost system, which can detect a fall and contact emergency services or a family member, without the need for the user to constantly carry a physical sensor, or have their privacy compromised.

Our project will implement a low-cost fall assistance system for the elderly. The proposed system will use a passive sensor to detect activity in a given room, such as a software defined radio, or infrared sensor. It is important to note that this is the only sensor that will be in use at all times. If activity is detected, then video taken from a low-cost DVS camera will be sent wirelessly to a central server. A DVS camera was chosen, as it only detects movement, and is considered less intrusive than a normal camera. The central server will then use deep learning to classify the activity, and determine if somebody has fallen. If a fall is detected, the system will then use a microphone and speaker, along with voice recognition, to speak to the person, and determine the appropriate action. If the user requests help, the system will be able to call the emergency services, or a named contact who can then assist the user. This is building on a previous project using the low-cost DVS approach and a project that detected falls and deployed a robot to help.

A dynamic vision sensor (DVS), instead of recording full frames only records the differences between those frames and so only moving objects are visible in the output. This lowers power consumption, reduces the of unused data being transmitted, and is perceived to be less intrusive because images taken don’t resemble people. In the previous low-cost DVS project, a camera was used that already had an interface developed for use with a Field Programable Gate Array (FPGA), this camera was inexpensive and not of optimal quality or framerate. As the ‘DVS’ camera works by differencing images, the framerate is very limited by the camera used.

Therefore, by creating a custom camera interface, a more optimal camera can be used. This should allow for a great improvement in framerate and overall performance of the DVS Camera.

An FPGA will be used to process the data from the camera, and generate the DVS images. This was chosen due to its highly parallel nature, meaning it can process the frames at a high rate. Ideally, a PYNQ FPGA will be used, due to its ability to support the full range of python image processing libraries. The frames obtained from the low-cost DVS camera must be wirelessly transmitted to a server where they will be classified. This type of data transmission is desired it means the embedded camera system could be placed anywhere in a house for home monitoring. Several methods to transmit the video feed wirelessly were identified such as: implementing a Software Defined Radio (SDR), connecting an FPGA directly to a Wi-Fi Router via Ethernet or using a PMOD WiFi Interface Module.

When the DVS data is received by the computer, it will be fed into a deep learning algorithm to determine whether a person has fallen over or if some other event has occurred e.g. a family pet running about. Deep learning was chosen as it has been used effectively in video classification applications. If possible, pre-trained networks will be used as a starting point to reduce the time taken to train the network. The initial stages of the project will involve researching existing work and determining which approach is best for the project. It will also involve obtaining an existing dataset to train and test the network if one exists, or creating one if a dataset cannot be sourced.

Once the classification has determined that there has been a fall, the final part of the system will determine whether or not the person requires assistance. Many of the elderly who fall do not need hospital admission. The voice recognition system aims to avoid unnecessary ambulance calls by asking the fallen person if they require an ambulance. If they do not need medical attention, asking if they need to call a named contact for help. If there is no answer or an unintelligible answer, an ambulance will be called. It is important to assume the worst-case scenario because the longer an elderly person is on the floor, the higher the risk of death [2]. Once the elderly person’s condition has been identified, the system will call the appropriate person.

Voice recognition can be implemented using Deep Learning, specifically Convolutional Neural Networks, Recurrent Neural Networks, or existing APIs such as the Google Speech API. As with the fall classification part of the project, this section will begin with extensive research and evaluation of existing methods and finding a dataset.It is important that this part of the project and the video classification are accurate as it is vital that the emergency services are not called when they are not needed.

Bibliography

[1] NHS, NHS, 24 April 2018. [Online]. Available: https://www.nhs.uk/conditions/falls/. [Accessed 10 October 2019].

[2] D. Wild, U. S. Nayak and B. Isaccs, “How dangerous are falls in old people at home?,” *British Medical Journal,* vol. 282, pp. 266-268, 1981.

B. Project Objectives:

Project objectives must be stated in such a way that they can be translated into achievable goals during the conduct of the project. For this reason, the stated objectives must be specific and realistic to be attained within the time provided. It is important to note that the achievements of the project work will be measured against the objectives stated here. Copies of this section will be made available to persons involved with the assessment of this project.

1. *Under the “Importance” column below, enter one of the following as appropriate: “Major", “Minor", or “Optional”.*

Project Objectives (add more if required)	Importance
Gain appropriate background knowledge of the problem, existing technologies, and possible solutions	Major
Develop FPGA/Camera system to replicate a DVS camera using a low-cost sensor	Major
Develop a method of detecting activity within the room, so that the system knows when to transmit data	Major

Detect activity in the room using a passive sensor	Optional
Transmit DVS data wirelessly securely to computer to be processed	Major
Use data from DVS camera to determine if a person has fallen	Major
Use a speaker and microphone to speak to the person, and determine if emergency services are required	Major
Allow the person to chose who will be contacted e.g. emergency services, relative, neighbour	Minor

PART 2:PROJECT WORK PLAN

Identify project milestones and summarise your work plans in the table below in the order you do them. (Example: preliminary design, prototyping, simulation modelling, results validation, write-up, etc).

	Project Milestones/Work Phases	**Expected Week Time**
1	Research/literature review into potential hardware, DVS cameras, deep learning for motion detection, and voice recognition	Week 5 to week 7
2	Prototype low-cost DVS system, wireless data transmission, fall detection, and voice recognition	Week 7 to beginning of semester 2
3	Complete interim report, detailing progress to date, and creating oral presentation	Week 10 to week 11
4	Finalise low-cost DVS system, wireless data transmission, fall detection, and voice recognition, ensuring that each part of the project will be compatible with the rest of the system	Sem 2 week 1 to 3
5	Combine sections into complete system, and troubleshooting any compatibility issues	Sem 2 week 3 to week 6
6	Test system to ensure correct functionality	Sem 2 week 3 to week 6

7	Prepare tradeshow demo, poster, video, and final report	Sem 2 week 6 to week 11

PART 3:RESOURCE REQUIREMENT

A. Software:

List the software required for the project. This includes programming languages, application packages, CAD tools, etc.

Software *(indicate version no. if applicable)*	Software Administrator *(CS Dept, EEE Dept, Comp. Centre)*	Target Use *(teaching, research, etc)*	Platform *(PC, workstation)*	Expected Usage *(hours/week)*
Matlab (with Simulink)	EEE department	Research and development	PC/own laptops	?
Python	EEE department	Research and development	PC/own laptops	?
GPU server	EEE department	Training of neural networks	PC/server	?
Vivado	EEE department	Research and development	PC/own laptops	?
Github	None	Source control and project management	PC/own laptop	?

B. Hardware:

List major hardware components such as circuit boards, LSI/VLSI integrated circuits, and special purpose components.

Computer capable of developing and training deep learning/machine learning; FPGA with PYNQ framework; camera; basic circuit components; transmitter/communications device; speaker; microphone

C. Laboratory/Work Area:

Indicate the laboratory room(s) and/or project work area for the project.

RC351, RC237/239

D. Logbook:

Confirmation that each student has A4, hardback, bound logbook.YES/NO (delete one)

PART 4:TECHNICAL RISK

Management of project work requires that technical risk be assessed in advance, during initial planning and as an ongoing process. As the first stage to this process, identify any aspects of risk associated with your project proposal. Risk in this context is taken to mean any event or action (or inaction) that would jeopardise any project outcomes or significantly impede project progress. Furthermore having identified such potential risks, indicate what actions you would take to mitigate the effects of this risk. (Consult your sponsor for advice but examples of such risks include non-delivery of a key component, illness or absence from University, non-completion by student or other of key deliverable, equipment malfunction, extended learning curves- new techniques or software, etc).

	Possible Risk:	**Mitigating Action:**
1	Sickness	Clear contact with team, proposer, and mentor if work progress may be comprimised
2	Loss of work	Constant back-up of work using services like the university H drive, GitHub, and USB devices
3	Hardware break/failure	New equipment acquired or old equipment repaired
4	Project too ambitious/learning curve too high	Scope reduced

Students will be asked to reflect upon parts 1, 2 and 4 in the interim reports and also in the final report.

PART 5:SAFETY DECLARATION

All project students must be aware of the need for safe working during the conduct of their project. The Area Safety Regulations for the Department of Electronic and Electrical Engineering, which appear in the Undergraduate Handbook, provide general guidance. Project students should consult with their Supervisor to obtain specific instructions or written Risk Assessment relating to their own project. By signing at the end of this form, the project student is declaring that:

1. he/she has attended the EEE UG safety seminar.
2. he/she has completed the online safety assessment quiz
3. he/she has read and understood the Area Safety Regulations and will abide by these regulations during the conduct of the project, and
4. he/she has consulted with the project supervisor who, if applicable, has specified the additional Risk Assessment which is as stated below:

Additional Scheme of Work Specified By Project Supervisor *(enter NONE if not applicable)*

…….

…….

…….

…….

Signature of Student:…………………………………………………………………

Signature of Student:…………………………………………………………………

Signature of Student:…………………………………………………………………

Signature of Student:…………………………………………………………………

Signature of Student:…………………………………………………………………

Signature of Student:…………………………………………………………………

Date…………………………………….

7.3 Glossary

Term	**Meaning**
ADC	Analogue to Digital Converter
ANN	Artificial Neural Networks
API	Application Program Interface

APU	Application Processing Unit
ASSP	Application-Specific Standard Product
AXI	Advanced Extensible Interface
CLB	Configurable Logic Block
CNN	Convolutional Neural Network
CPI	Camera Parallel Interface
CPU	Central Processing Unit
CSI	Camera Serial Interface
DAC	Digital to Analogue Converter
DNN	Deep Neural Networks
DSP	Digital Signal Processing
DVS	Dynamic Vision Sensor
FEC	Forward Error Correction
FPGA	Field Programmable Gate Array
HDL	Hardware Description Language
HREF	Horizontal reference Signal
IC	Integrated Circuit
IIC	I^2C
IO	Input-Output
IP	Intellectual Property
MIPI	Mobile Industry Processor Interface
OS	Operating System
PCLK	Pixel Clock
PL	Programmable logic
PMod	Peripheral Module
PS	Processing System
PYNQ	Python productivity for Zynq
RF	Radio Frequency
SDR	Software Defined Radio
SoC	System on Chip
VSYNC	Vertical Sync Signal

WLAN	Wireless Local Area Network

7.4 Code

All project code and other materials such as the bill of materials, Gantt chart, and system-level design can be found on the project GitHub [90].

7.5 Python Library Dependencies (So far)

All libraries were obtained using pip install to ensure all the correct dependencies were obtained.

7.5.1 For use in machine learning

Library Name
OpenCV
OS
NumPy
TensorFlow

7.5.2 For use in voice recognition

Library Name
PyDub
Matplotlib
SciPy
OS
Wave
PyAudio
SciKit-Learn
gTTS
PyGame
SpeechRecognition
PocketSphinx
Swig