Department of Informatics
Computer Science

Interim report 2019

Robot Control Interface
From Arduino to ARM Mbed

Contents

1. Introduction

Arduino is a prevalent platform in its use amongst hobbyist, but not so much amongst professional engineers. Its popularity with hobbyists is largely due to the lack of required knowledge to get started, due to its extensive official software libraries and example code supported with further such code written by the online community. This is why it is a platform that is still widely used to drive plenty of devices and applications.

Unfortunately, the problem for developing real products in Arduino is that it is not free for commercial use and even requires all object files to be made public. The Arduino boards also have numerous other limitations that make them not quite as preferred by professionals, as alternate controllers that are Arduino-compatible can be more powerful due to their increased RAM, Flash Memory and Clock Speed; widening the applications that can be supported. It is becoming increasingly popular for devices currently running under Arduino to be used within IoT applications, as exemplified with a hackster.io tutorial on using an Arduino MKR ENV Shield with the Arduino IOT Cloud and Amazon Alexa to interact with the board's sensors [1]. However, other boards have better IoT functionality from features including Bluetooth, sdCard and WiFi abilities that are present on platforms such as Mbed. The Mbed solution has more potential for growth in the long-term and is more suited for the future of IoT, providing an excellent framework to develop cloud-connected devices.

One can switch to higher performing Arduino boards such as Arduino Mega 2650 with 8KB of RAM and 256KB of Flash Memory, but this is still not as high as certain Mbed boards and can have bigger and bulkier Printed Circuit Boards (PCBs), making it difficult to fit them inside certain machines.

As it appears moving to higher functioning boards will be useful for this increasingly IoT-based age, the problem one must consider is how to retire and replace previously used out-dated hardware, and porting the implementations and valuable software solutions that were running on them to another better platform. The motivation of this project will be to explore this process.

2. Objectives

The ZUMO robot for the Arduino platform is well-established, but although ports have been created for certain ARM Mbed platforms (e.g. the FRDM-KL25Z) there are currently none specifically for the FRDM-K64F. Arduino UNO R3 is a microcontroller board based on the ATmega329P microcontroller unit (MCU) [2], while the FRDM-K64F contains the power efficient Kinetis K64F MCU [3].

The purpose of this project will be to remove the Arduino UNO R3 microcontroller board and translate its library codes (ZumoShield) [4] to create a C++ library that runs on the more powerful ARM Mbed FRDM-K64F microcontroller board. It can then be used to control the ZUMO robot and its components, sensors and actuators (LEDS, Pushbuttons, Motor Drivers, Buzzer, Infrared Reflectance Sensors, Accelerometer, Magnetometer and 3-axis Gyroscope). The final report will assess the differences in the robot's performance and functionality on both the platforms and evaluate the process and challenges faced when "software porting".

3. Background and Requirements Analysis

3.1 Primary Tasks

3.1.1. Code translation

The Arduino UNO R3 and FRDM- K64F MCUs are both programmed in C++ so the specific translation of code is quite achievable due to the similarities in the languages. The ZumoShield library will need to be translated, including its header files(.h) and source files(.cpp). The header files are interface-like while the source files provide an implementation of key features of the Zumo robot that are essential for its usage. Table 1 shows the header and corresponding source files (if any) for the ZumoShield library files that must be ported for use on the FRDM-K64F.

Header files(.h)	**Source files(.cpp)**
L36.h	L3G.cpp
LSM303.h	LSM303.cpp
PololuBuzzer.h	PololuBuzzer.cpp
Pushbutton.h	Pushbutton.cpp
QTRSensors.h	QTRSensors.cpp
ZumoMotors.h	ZumoMotors.cpp
ZumoBuzzer.h	
ZumoReflectanceSensorArray.h	
ZumoShield.h	

Table 1 ZumoShield header and source files

Additionally, the example code within the library will also need to be translated as they contain control over features that will be vital for comparing the performance and functionality of the two microcontroller boards. The example code files(.ino) are listed below.

- BorderDetect
- Compass
- LineFollower
- MazeSolver
- PushbuttonExample
- QTRAExample
- QTRARawValuesExample
- QTRRCExample
- QTRRCRawValuesExample
- RCControl
- SensorCalibration
- SumoCollisionDetect
- ZumoBuzzerExample
- ZumoBuzzerExample2

- ZumoBuzzerExample3
- ZumoMotorExample

3.1.2. Software Porting Investigation

The main focus on the programming end will be to explore the concept of "software porting"; investigating the process of adapting software for use on an alternate platform to the one it was originally designed to execute on. I.e. altering software for it to be usable in different environments. "Embodiments of the present invention provide a method of porting software. The method of porting Software comprises receiving a set of code operable in a first operating environment. The method further comprises converting the set of code into a class. The method further comprises providing the converted set of code operable in a second operating environment [5]." Notable solutions, findings and challenges during the process of software porting will be documented and concluded in the final report.

❖ *Language considerations*

Arduino runs machine code that has been compiled from the higher languages of either C or C++; hence the ZumoShield library contains C++ files. At the same time, many of the example programs in the library are written in C as the Arduino UNO R2 has significantly less features and components in contrast to the FRDM-K64F, and C is a simpler language. This means the ZumoShield library is currently less likely to be utilising object-oriented features of C++ in the Arduino environment, which will be required in order to test the additional functionality of the FRDM-K64F. The code will need to be translated in Mbed to a format that more closely mirrors C++ programming.

One should firstly consider the design methods best supported by both languages in particular — C follows a top-down approach while C++ follows bottom-up programming. The paradigms for C and C++ are imperative (structural) and object-oriented respectively; C supporting a structure with more computational steps and C++ focusing on hierarchies. More design is required in the development of an imperative language: therefore translating to C++ will be easier than the reverse. The benefit of the increased use of objects will also make the library better organised, taking advantage of the additional features of C++. The most essential aim is to be aware of the native capabilities of both languages in order to focus on the best way to execute the desired functionality instead of focusing on the manner in which it was originally coded.

Others have already evaluated the process of translating code from an Arduino MCU to an ARM Mbed MCU. Considering their discussions and findings will be of considerable use for being cognisant of important aspects of software porting before commencing the project. One such investigation includes the porting of software from an Arduino Mega 2560 R3 to Mbed NXP LCP1768 [6]. The author of this s-lab article states that the code is much cleaner working in Mbed, describing code on Arduino to be chunky as most programmers will situate all of their code in one file. The reason for this is due to library creation being notably arduous in Arduino, as is the debugging process. The author goes on to explain that Arduino does not exploit commodious Object-Oriented Programming practices; ergo C++ on Mbed is cleaner and more modular. This element is a necessity for team work as each team member can work on their own module in their separate set of files.

	Arduino Mega 2560 R3	Mbed NXP LPC1768
Price	€35	€46.96
Flash Memory	256KB	512KB
Clock Speed	16MHz	96MHz
RAM	8KB	32KB
EEPROM	4KB	Entire flash memory can be used
Pins	54	40

Table 2 Tech specs [7] (data updated by author to match with current information)

The author of the s-lab article also draws attention to the advantages of the Arduino online community. They highlight that there is an assortment of various implementations for the same endeavoured outcome of a program required on the Arduino Software (IDE) platform, a key advantage of most open source technologies. Unfortunately, these code examples generally work but can frequently be poorly coded, contain copious bugs, or be rather unoptimised even if they are found directly on the Arduino website. Alternatively, the online Mbed community is significantly smaller but nevertheless consists of more advanced programmers, exemplified in the provided FRDM-K64F Freedom Sensor Example creating "eCompass" equivalent to the already pre-defined ZumoShield "Compass" example for Arduino, by using the on board 6-axis sensors [8]. He advises that the Mbed Handbook [9] and Cookbook [10] provided can be of valuable assistance, enabling one to import code directly into the IDE for testing, yet lacking in the sheer selection of code examples potentially required for one's project that are abundant in Arduino. This deficiency can still be counteracted due to the fact that C++ is one of the most utilised languages at this juncture, making it possible to discover one's preferred means of completing a task despite it requiring extra exertion in contrast to more relevant online solutions.

Yiu, author of 'The definitive guide to the ARM Cortex-M0*'* and staff engineer at ARM Ltd. explicates, "In 8-bit and 16-bit microcontroller programming, the peripherals control is usually handled by programming to registers directly. When using ARM microcontrollers, many microcontroller vendors provide device driver libraries to make use of the microcontroller easier. You can use these library functions to reduce software development time or write to the hardware registers directly if preferred. If you prefer to program the peripherals by accessing the registers directly, it is still beneficial to use the header files in the device driver library as these have all the peripheral registers defined and can save you time preparing and validating the code." [11]

❖ *Programming Environments*

Additionally, one must consider the key similarities and differences in the programming environment for the two microcontrollers as their structure is developed to support the unique functionality of both boards. These environments are provided by Arduino and ARM Mbed, namely the Arduino Software (IDE) supporting any Arduino board [12] and the Mbed OS doing the same for Mbed boards. "The Arduino Uno is programmed using the Arduino Software (IDE), our Integrated Development Environment common to all our boards and running both online and offline" [13]. Once the board is specified both IDE's include built-in support for them, Arduino using the "Board Manager". This board manager "sets the parameters (e.g. CPU speed and baud rate) used when compiling and uploading sketches; and sets the file and fuse settings used by the burn bootloader command" [14]. The FRDM-K64F is "fully supported in the mbed platform, so it gets access to the free tools and SDK that provides experienced embedded developers with powerful and productive tools for building proof-of-concepts" [5].

The main similarity between the two IDEs is that some of the functions in Mbed also have similar names to Arduino functions such as DigitalIn and DigitalOut, making the process of software porting smoother.

The Mbed platform contains unique features that enable it to be a basis for low-power systems, such as the RTOS library that allows one to run multiple independent threads making the code design much simpler. Rather than having one big main loop that handles all the implementation, the code for separate tasks can be split into threads. The Mbed OS also has Cloud capabilities that are already integrated, allowing one to seamlessly transfer data from the device to the cloud and aggregate it using one of ARM's cloud partners. It maintains a TCP/IP stack and Socket API on top of multiple transports (Ethernet, WiFi and 3G) [15] and a scheduler for ensuring tasks get their appointed run-time. It also includes a hypervisor, namely uVisor (micro-visor), to separate different tasks so that user-tasks cannot compromise the security of the complete system. This capability also opens the possibility of running user generated or insecure code on the devices, as the uVisor will ensure that it only takes granted resources and cannot access secure information. It also protects against attacks, as if someone breaks into a complicated RF stack on one's system, they will find it difficult to gain access to the rest of the system. Soon, uVisor will be "superseded by the Secure Partition Manager (SPM) defined in the ARM Platform Security Architecture (PSA). uVisor is deprecated as of Mbed OS 510, and being replaced by a native PSA-compliant implementation of SPM" [16]. These features are critical for IoT applications. The Mbed OS also provides its own APIs which allow code to work on different microcontrollers in a uniform way. This reduces a lot of the challenges in getting started with microcontrollers [17].

Arduino does not contain a debugger for checking scripts. On one hand this is suitable for inexperienced developers but on the other an experienced developer would much rather have a debugger; this missing feature often being an impediment to Arduino being taken seriously. Similarly, the Mbed OS online IDE also has this feature missing, but alternatives lie in the alternative desktop IDEs available such as the ARM Mbed CLI (a Python-based tool) and ARM Mbed Studio (which is still in public beta mode, still taking feedback from alpha testers).

The author of the s-lab article [6] determines that albeit the simplicity of the Arduino Software (IDE) on the basis of its handful of features, it is dependable and well-established. They instead liken the Mbed OS online IDE to a platform that is in a somewhat more beta stage of development rather than a finalised product. This is demonstrated through instances such as a high potential for crashes of the IDE to occur if the internet connection is lost, and goes on to propose that perhaps it is worth taking a look at alternatives to the online IDE to overcome this issue. Nevertheless, they point out a crucial component of the Mbed OS online IDE which is its built-in SVN capacity. This is efficacious in enabling a team of programmers to work collectively on the same project, "…keep various versions of the code and merge it all in a clean way." They conclude that the Arduino Software (IDE) is markedly more solid in regards to reliability, while the Mbed OS online IDE is imperative for large projects with prodigious amounts of code and multiple contributors.

Lim presents another perspective that shares a likeness to the one above, expounding that the online IDE lacks straight-forwardness and is laborious when attempting to determine "how the pins are defined for each board" [18]. Likewise he disapproves of the platform's dependency on an internet connection, stating, "…I don't want to necessarily depend on it to compile my code." He goes on to suggest PlatformIO as an offline option, "which is a simplified interface to embedded development, overlaid over other IDEs like Atom or Microsoft Visual Studio Code... It is also less cumbersome than the mbed drag-and-drop method." Mbed uses a custom boot loader and appears as a USB flash drive when plugged into the computer via the USB port. One must then compile their code using the online IDE, download the .hex file and copy it to the Mbed, resulting in time consumption. Both of the above viewpoints can be solved by using ARM Mbed CLI or ARM Mbed Studio.

Styger's experiment is one closely related to the one going to be carried out in this project. He programmed the FRDM- KL25Z to be usable with the Zumo robot instead of an Arduino board in the CodeWarrior IDE using Processor Expert [19]. NXP, creators of Freedom boards state, "Processor Expert® technology is a development system to create, configure, optimize, migrate, and deliver software components for our silicon. This technology is integrated into NXP's CodeWarrior® products supporting S08/RS08, S12(X), Coldfire, Coldfire+, Kinetis®, DSC 56800/E, and some Power Architecture processors. Processor Expert technology makes it much easier for you to deal with the low level intricacies of a hardware platform in an optimal manner. You do not have to accept generic one-size-fits-all drivers. You design custom peripheral drivers ideally suited to your needs, without having to know everything about the hardware" [20]. However, Processor Expert has been replaced by its successor, the MCUXpresso Configuration Tools, since Styger published his article in 2013. The NXP official Getting Started with the FRDM-K64F guide lists the MCUXpresso Software Development Kit (SDK) + IDE and Zephyr™ OS as recommended development paths, of course accompanied by ARM Mbed [21].

Finally, one gets no real experience of professional development tools with Arduino. If one starts their journey of micro-controllers with Arduino then it will becomes increasingly difficult to make the complex intelligent circuitries in future. The easy to use hardware/software of Arduino prohibits a person to learn the basics of many things like Serial communication, ADC, I2C etc. Porting over to FRDM-K64F is a useful learning process for any engineer if they would like a deeper understanding of circuitry.

❖ *Hardware*

The hardware itself must also be compared, primarily the increased RAM, Flash Memory and Clock Speed available on the FRDM-K64F. This will enable one to run all libraries and calculations without the crashing that is often caused by high amounts of code on the Arduino UNO R3, causing the code written and run on the FRDM-K64F to be an easier process when software porting. This is due to being able to avoid wasted time and enabling one to not be too closely concerned with the amount of code they are writing.

One can then consider the pin configurations which differ marginally between the microcontroller boards. The pinout diagrams show the layout of the pins (which make up ports); the Arduino UNO R3 containing 28 pins and FRDM-K64F containing 64 shown in Figure 1. Figure 2 illustrates, "…the mapping between Arduino pins and ATmega328P ports" [2]. To elucidate, these board pins correspond to the component leads extending from each of the four sides of the ATmega328P, depicted in the pinout diagram in Figure 3. On the other hand, "The freedom board headers enable up to 64-pins and give access to most of the Kinetis K64F signals. Outer row pins deliver right signals to meet Arduino R3 standard, the inner row is connected to up to 32 additional Kinetis K64F pins" [3]. Figure 4 indicates the Kinetis K64F MCU signal connections with the board components (RGB LED, Motion Sensors) and extension sensors (SD Card, Bluetooth, WiFi). The named pins in this diagram correspond to just a few pins of the 144 total pins in the Kinetis K64F MCU, presented in Figure 5. The pins on each board are laid out in a similar manner particularly with the 32 outer pins of the FRDM-K64F, due to it containing, "Form-factor compatible with the Arduino® UNO Rev3 pin layout" [22]; which will greatly assist in understanding the parallels between the pin signals in order to match and adapt functionality from one board's pins to another: easing the process of software porting. The crucial distinction in regards to the hardware is the additional functionality provided by the 32 extra inner pins of the FRDM-K64F. That being said, this will not have any implications on the software porting itself, but more on other requirements set out in this report such as testing the difference in the functionality of the Zumo robot between the two boards by writing further code that will utilise the potential extra functionality from the additional pins.

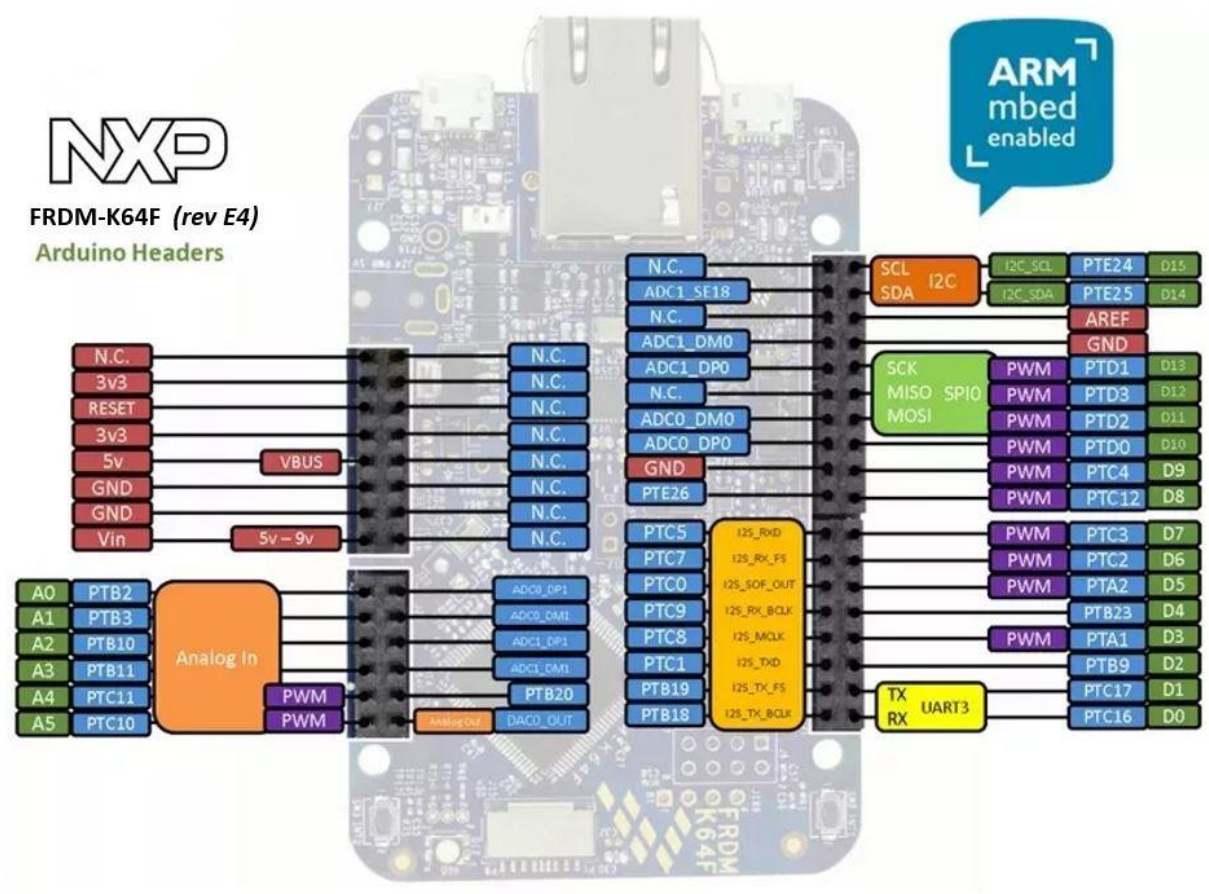
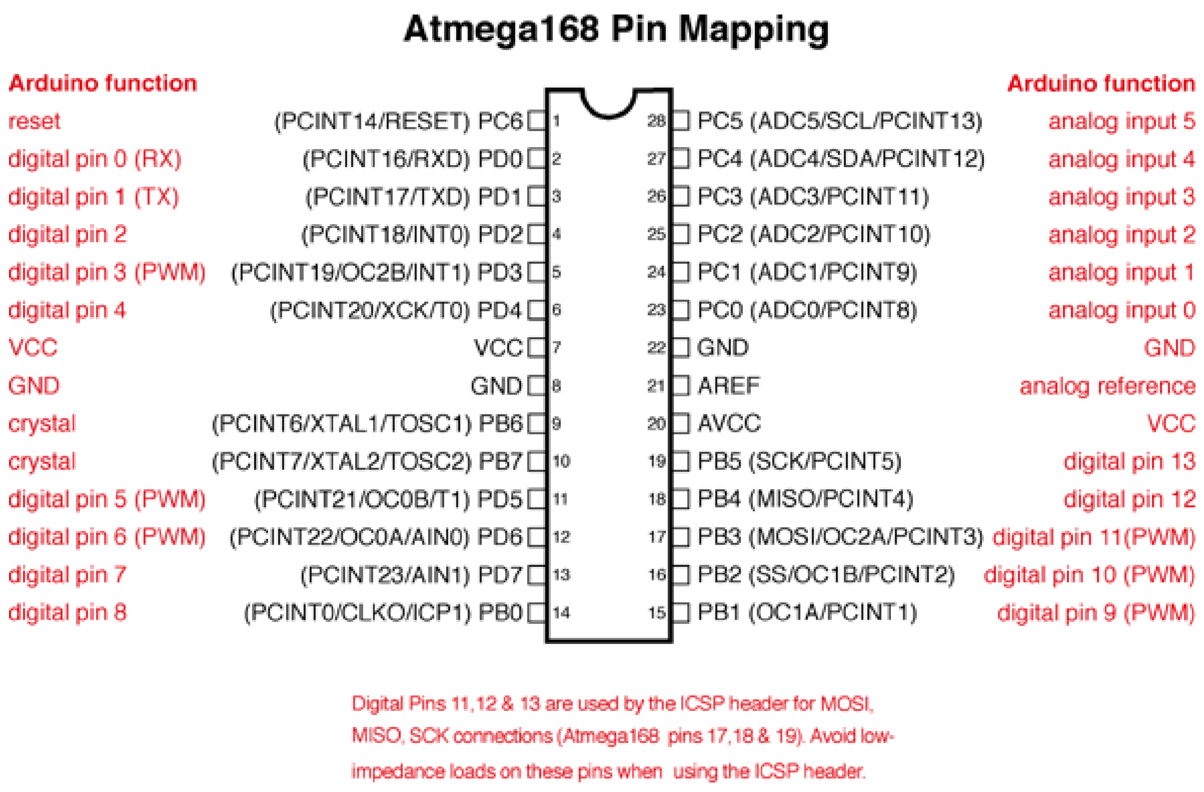


Figure 1 Arduino and NXP Header Pinout [3]



Atmega168 Pin Mapping

Arduino function					Arduino function
reset	(PCINT14/RESET) PC6	1	28	PC5 (ADC5/SCL/PCINT13)	analog input 5
digital pin 0 (RX)	(PCINT16/RXD) PD0	2	27	PC4 (ADC4/SDA/PCINT12)	analog input 4
digital pin 1 (TX)	(PCINT17/TXD) PD1	3	26	PC3 (ADC3/PCINT11)	analog input 3
digital pin 2	(PCINT18/INT0) PD2	4	25	PC2 (ADC2/PCINT10)	analog input 2
digital pin 3 (PWM)	(PCINT19/OC2B/INT1) PD3	5	24	PC1 (ADC1/PCINT9)	analog input 1
digital pin 4	(PCINT20/XCK/T0) PD4	6	23	PC0 (ADC0/PCINT8)	analog input 0
VCC	VCC	7	22	GND	GND
GND	GND	8	21	AREF	analog reference
crystal	(PCINT6/XTAL1/TOSC1) PB6	9	20	AVCC	VCC
crystal	(PCINT7/XTAL2/TOSC2) PB7	10	19	PB5 (SCK/PCINT5)	digital pin 13
digital pin 5 (PWM)	(PCINT21/OC0B/T1) PD5	11	18	PB4 (MISO/PCINT4)	digital pin 12
digital pin 6 (PWM)	(PCINT22/OC0A/AIN0) PD6	12	17	PB3 (MOSI/OC2A/PCINT3)	digital pin 11(PWM)
digital pin 7	(PCINT23/AIN1) PD7	13	16	PB2 (SS/OC1B/PCINT2)	digital pin 10 (PWM)
digital pin 8	(PCINT0/CLKO/ICP1) PB0	14	15	PB1 (OC1A/PCINT1)	digital pin 9 (PWM)

Digital Pins 11,12 & 13 are used by the ICSP header for MOSI, MISO, SCK connections (Atmega168 pins 17,18 & 19). Avoid low-impedance loads on these pins when using the ICSP header.

Figure 2 ATmega328P - Arduino Pin Mapping [23]

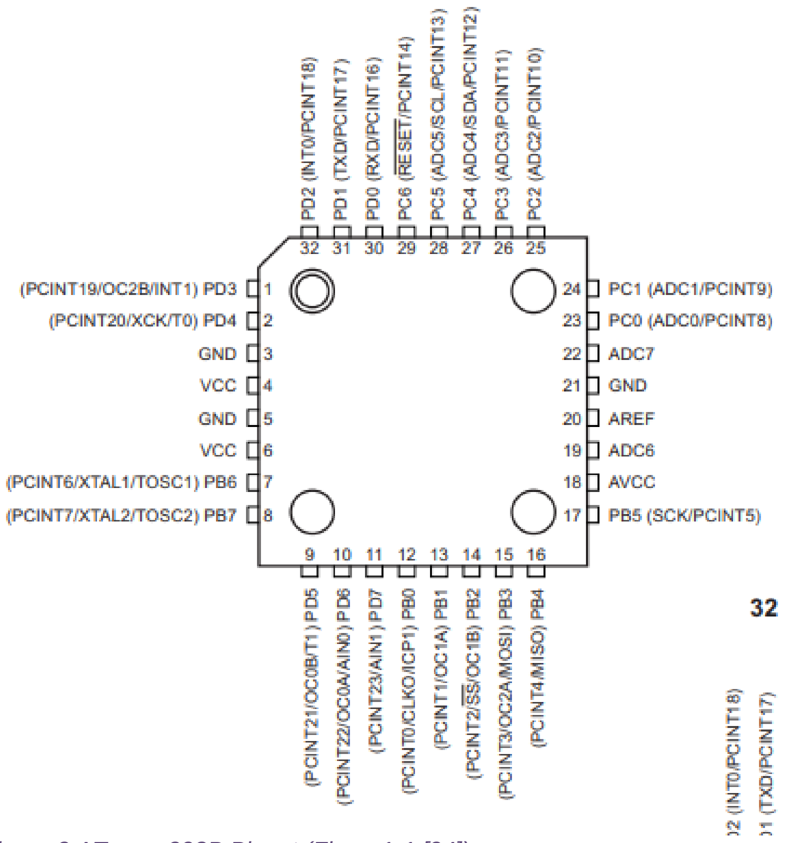


Figure 3 ATmega328P Pinout (Figure1-1 [24])

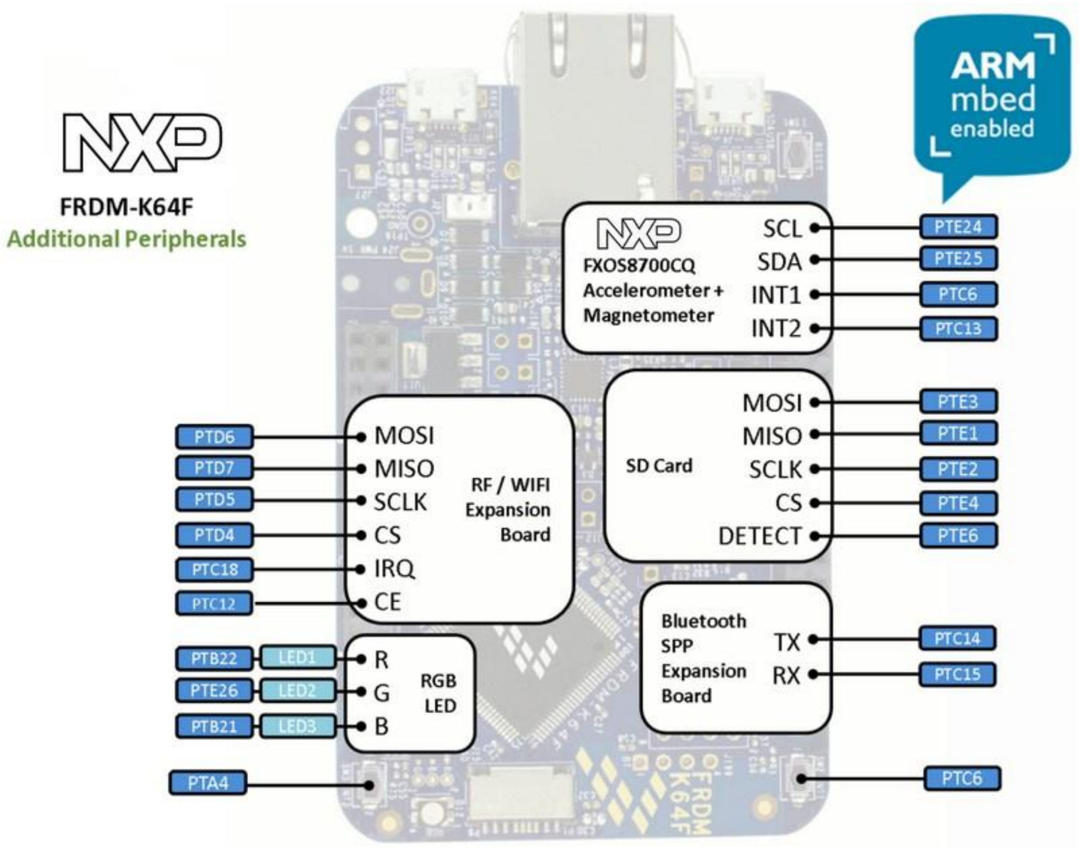


Figure 4 FRDM-K64F Additional Peripherals [3]

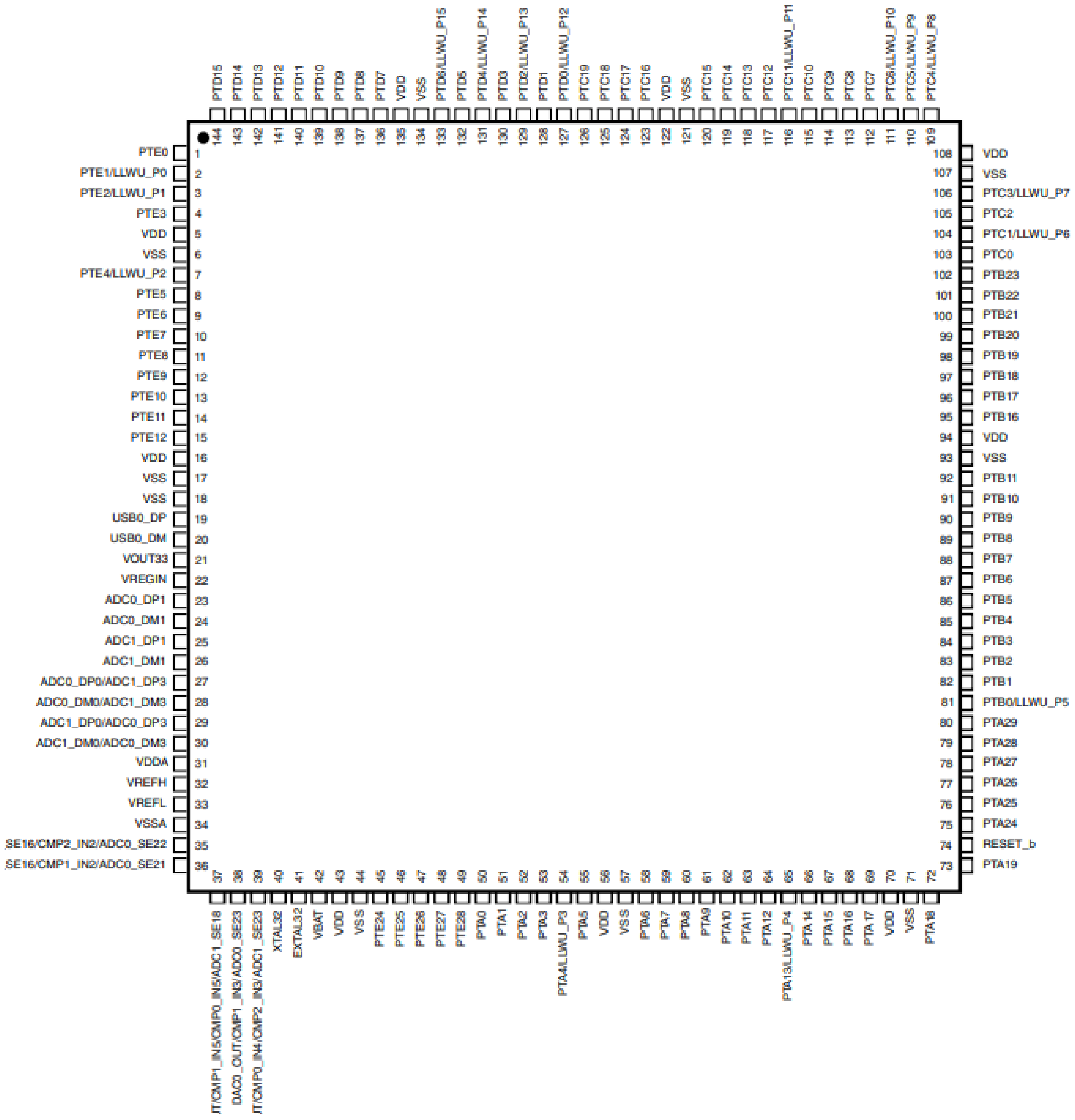


Figure 5 144 LQFP K64 Pinout Diagram (Figure 37 [25])

During Styger's process, he noticed that, "The Freedom board does not generate 5V from the V_IN (Battery) supply voltage. The Zumo Arduino shield expects the 5V generated from the Arduino microcontroller board" [26]. To overcome this limitation, he used a voltage converter in order to generate 5V from 2.7..11.8V by soldering it on a 3 pin header and placed it on the Zumo shield. Fortunately, this issue will not have to be dealt with in order to software port to the FRDM-K64F board, as it has, "Board power-supply options (onboard 5 to 3.3V regulator):

- USB Debug 5V
- USB Target 5V" [3]

Styger also took note of strategic tactics he could implement to make the debugging process more convenient, advising, “Generate bread crumbs for debugging: used the command line shell built-in, transmitted messages over Bluetooth, or using an extended USB cable”. For this maze experiment he also warns to, “Never underestimate problems with sensors. Depending on the laser printer and paper used, the reflections were different…”, causing misperception on the cause of the occasional sensor unreliability [27]. This was also due to the use of laser printing to create the black lines for the Zumo robot to follow instead of black tape which could hypothetically cause less reflection.

Overall it would be worth noting in the final report the general, Arduino or Mbed specific, and Arduino UNO R3 and FRDM-K64F microcontroller board specific challenges and significant phases of the process of software porting.

3.1.3. Microcontroller board performance comparison

Table 3 indicates the technological specifications for the two boards, the data highly indicating that the FRDM-K64F would yield faster results and be generally more powerful with potentially superior functionality. The increased RAM, Flash Memory, and Clock Speed will cause the code to be run without crashing, and the faster frequency and baud rate will give better accuracy of data and more control of the robot. The final requirement of the project will be to investigate and test the boards to see if these assumptions hold true.

	Arduino UNO R3	FRDM-K64F
Price	€20	€31.64
Flash Memory	32KB	1024KB
Clock Speed	16MHz	120MHz
RAM	2KB	256KB
EEPROM	1KB	Entire flash memory can be used
Pins	28	32 (64 including inner and outer)

Table 3 Tech Specs for Arduino Uno R3 & FRDM-K64F

A list is presented below depicting examples of types of experiments that will be carried out in order to fulfil the investigation.

Concepts evaluated when comparing boards:

- Timing
  - ⇒ Compile time - indicates perhaps which IDE or compiler is more efficient for executing software and therefore which board might be faster due to its IDE
  - ⇒ Time Complexity of programs on either board – computational steps
  - ⇒ Robot response time
- Functionality – the variety of useful functions that can be implemented and directly compared for each board
  - ⇒ FRDM-K64F’s 6-axis combo Sensor Accelerometer and Magnetometer [28] vs. Arduino UNO R3 with just the sensors provided on Zumo
  - ⇒ Whether the additional inner 32 pins on the FRDM-K64F board can enhance functions performed by the Zumo robot in contrast to with the Arduino board
- Quality of performance i.e. error-level e.g:
  - ⇒ driving 10cm, which board executed this distance most accurately
  - ⇒ compare performance of buzzer on both boards
- Comparing the quality of execution of the sensors that come with ZUMO incl. [29]

    - ⇒ Quadrative and inertial encoders (accelerometer & gryo – motor and robot following straight line without interference)
    - ⇒ Inertial Measurement Unit
    - ⇒ Reflectance sensor array (infrared QTR sensors) e.g:
        - ➢ to test how accurately it can follow a straight line [26]
        - ➢ to test which board performed better at border detection
        - ➢ using the compass module to compare the performance of the robot coordinating 90 degree turns and driving in squares
        - ➢ using the compass module to compare the collision detecting ability

3.2 Secondary Tasks

Objectives that would appear further than the time limit include:

- Researching online user-defined Arduino examples that do not exist in the ZumoShield library for additional functionality of the Zumo robot, and porting that over to FRDM-K64F in order to further test the difference between the performance of the two boards
- Creating original functions that do not exist in the ZumoShield library with the same behaviour for additional functionality on the Arduino UNO R3 and FRDM-K64F boards, in order to further test the difference between the performance of the two boards
- Creating original functions to portray functionality that is unique to the FRDM-K64F board in order to document what abilities the Zumo robot is capable of on this board in contrast to Arduino UNO R3
    - ⇒ This includes the use of FRDM-K64F's RGB LED, SD Card, Bluetooth, Wifi that is not available on the Ardunio UNO R3
    - ⇒ The additional 32 inner pins on the FRDM-K64F have connections to these components. Also valuable to test what other signals and capabilities these pins uniquely provide to the Zumo robot
- Program the reflectance sensor array on the Zumo to solve a line maze on both boards and compare the performance on either [30] [27]
- Connecting an RC receiver to both boards and turning the Zumo into a radio-controlled vehicle, again comparing the performance
- Connect an RC servo to the Zumo Shield and control it with both boards, once again comparing performance

Styger offered his own improvements and extensions for his implementation of the Zumo robot solving a line maze, of which these were of interest for being potential extensions for this project:

1. Ability to return from the finish to the start automatically.
2. Exploring the full maze to find the smallest path.
3. Using dynamic speed: running faster on the long straight lines, while slowing down near the intersections. [27]

The extension would also explore additional sensors that can be purchased for less than $5 and comparing the performance of both boards with these sensors:

- Sensors for ZUMO e.g. lidar or sonar range finders, and sharp distance sensors [31]
    - ⇒ Necessary to purchase, "Connector and jumper wires, for connecting additional sensors and components" [29]
    - ⇒ How well or quickly it can detect nearby objects

- Sensors that can be attached to the Arduino UNO R3 and FRDM-K64F boards incl: temperature & humidity sensor, light sensors, collision sensors [3][1]
- Other added modules incl. vibration, joystick, digit display [3][1]
- Observing if the boards differ in performance when combining sensors with other functionality

3.3 Requirements Specification

A summary of the mandatory and optional requirements of the software implementation will now be presented below as a result of the above requirements analysis.

Mandatory Requirements	**Justification**
Translate L36.h and L36.cpp from Arduino for use on FRDM-K64F	Interfaces with and enables reading of raw data from 3-axis gyros
Translate LSM303.h and LSM303.cpp from Arduino for use on FRDM-K64F	Interfaces with and implements compass and accelerometer
Translate PololuBuzzer.h and PololuBuzzer.cpp from Arduino for use on FRDM-K64F	Enables the buzzer
Translate Pushbutton.h and Pushbutton.cpp from Arduino for use on FRDM-K64F	Enables the Pushbutton
Translate QTRSensors.h and QTRSensors.cpp from Arduino for use on FRDM-K64F	Implements the reflectance sensors
Translate ZumoMotors.h and ZumoMotors.cpp from Arduino for use on FRDM-K64F	Implements motor control
Translate ZumoBuzzer.h from Arduino for use on FRDM-K64F	A trivial subclass of PololuBuzzer
Translate ZumoReflectanceSensorArray.h from Arduino for use on FRDM-K64F	Provides an interface for using a Zumo Reflectance Sensor Array and provides access to the raw sensor values as well as to high level functions including calibration and line-tracking
Translate ZumoShield.h from Arduino for use on FRDM-K64F	A subclass of many of the above
Translate example code BorderDetect from Arduino for use on FRDM-K64F	Example code for detecting a white border on a black surface
Translate example code Compass from Arduino for use on FRDM-K64F	Example code for making use of the compass features
Translate example code MazeSolver from Arduino for use on FRDM-K64F	Example code for implementing the solving of a maze
Translate example code PushbuttonExample from Arduino for use on FRDM-K64F	Example code for implementing the use of the pushbutton
Translate example code QTRAExample, QTRARawValuesExample, QTRRCExample, and QTRRCRawValuesExample from Arduino for use on FRDM-K64F	Examples of code for the sensors
Translate example code RCControl from	Example code for the optional RC Control

[1] Sensors provided by SEEED studios

Arduino for use on FRDM-K64F	
Translate example code SensorCalibration from Arduino for use on FRDM-K64F	This example is in order to calibrate the Reflectance Sensor Array with its surroundings
Translate example code SumoCollisionDetect from Arduino for use on FRDM-K64F	Example code to test the robot's collision detection
Translate example code ZumoBuzzerExample, ZumoBuzzerExample2 and ZumoBuzzerExample3 from Arduino for use on FRDM-K64F	Example code in order to test the buzzer
Translate example code ZumoMotorExample from Arduino for use on FRDM-K64F	Example code in order to test the motor and drive the robot

Table 4 Requirements summary of primary and extension objectives

Optional Requirements
Porting user-defined Arduino examples of Zumo robot over to FRDM-K64F
Creating original examples that complete the same task for both Arduino UNO R3 and FRDM-K64F
Creating original examples that only support FRDM-K64F specific functionality e.g. RGB LED, SD Cards, Bluetooth, WiFi, and other FRDM-K64F pin signals
Code line maze solving for the Zumo on Arduino and Mbed
Improve line maze solving so Zumo can return from finish to start automatically, find maze shortest path, and use dynamic speed
Connect RC receiver to Arduino UNO R3 and FRDM-K64F
Connect RC servo to Zumo robot
Connect additional sensors to boards

4. Professional Considerations and Ethics

This project meets all the specified ethical requirements outlined in the Ethical Compliance Form attached in the appendix, as above all it will not involve any other members of the public save for this report's author and supervisor Ron Grau. The health, privacy, security and wellbeing of the two of us and the environment will not be impacted by the investigations on the Zumo robot as it has a negligible and inconsequential effect on its surroundings. This is due to the fact that it does not have access to or collect personal data from either of us, nor does it have sharp enough edges or a substantial power supply to cause physical harm or injury as "the Arduino's regulated 5V and 3.3V voltages supply power to the motor driver logic, buzzer circuit, and compass module on the Zumo Shield" [29]. The experiments will be conducted in a contained environment comprising of a small black surface such as a Dohyo (a wooden battle arena for Robot Sumo) portrayed in Figure 6 below or white sheets of paper as shown in Figure 7. As a result neither discrimination nor malpractice against Third Parties can occur, participants will not be exposed to any risks greater than those encountered in their normal working life and all relevant legislative and regulatory requirements will be met.

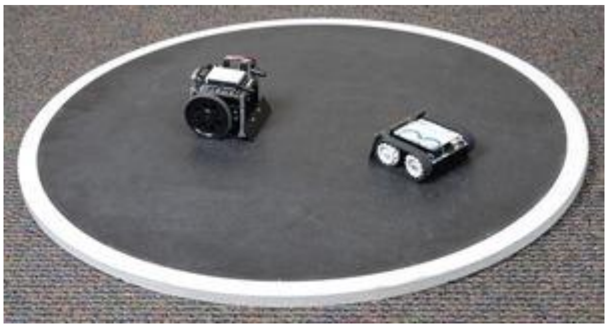


Figure 7 A Zumo Robot in a test arena for evaluating movement and IR sensors [29]

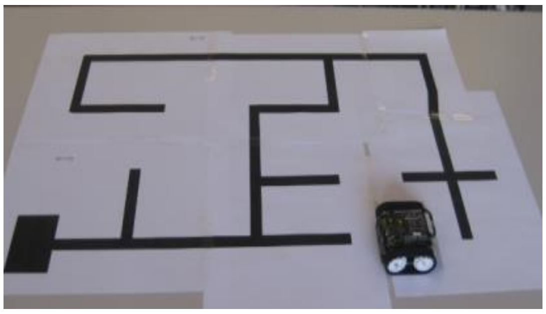


Figure 6 Maze for Zumo Robot [27]

The study materials comprise of software running on standard hardware, as indicated via the use of the the well-established ZumoShield robot, Arduino UNO R3 and FRDM-K64F boards.

The work undertaken in this project is within the professional competence of the author as it utilises an array of expertise provided by the author's Computer Science undergraduate degree, including the training of C++ during their foundation year. The author's professional knowledge will continue to be developed for the duration of the project due to the length of the degree course. The author will also make it a priority to independently find answers to fill gaps in their knowledge and will ensure any further research and references provided in the final report are up to date. Furthermore, the value of the extensive insight possessed by the author's supervisor will also be an aid in maintaining awareness of any technological updates, developments or standards that may be relevant and important for the investigation. The author will also have the opportunity to obtain alternate viewpoints and criticism from the supervisor which will be respected and always taken into account. Conclusively having discussed this in length with given supervisor, it has been agreed that no ethical review or ethical compliance form will be required.

6. Project Management

6.1 Project Plan

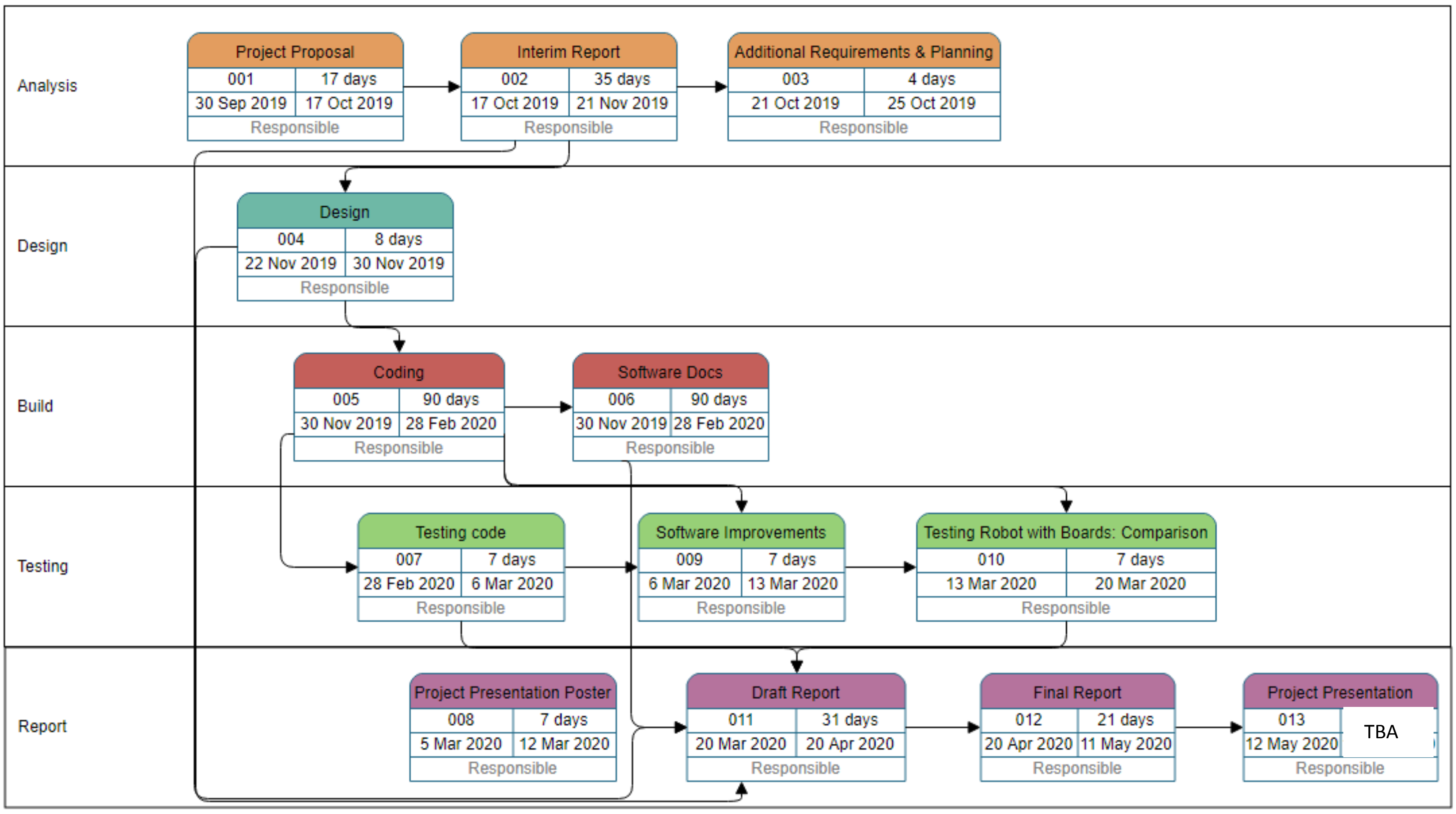


Figure 8 PERT chart

Figure 8 presents the time estimates of the project's tasks and phases, with the interdependencies represented by arrows, and start and finish date indicated in each box. The project proposal and Interim Report are the completed tasks so far, with coding, software documentation, code testing, project presentation poster creating, software improvements and robot experimenting occuring in the second term. The arrows in the diagram above indicate the interdependency of the tasks.

- Project Proposal: The creation of this document enabled the author to take the necessary steps to form an initial plan for the project, along with some necessary initial background research.
- Interim Report: The creation of this report enabled high-level planning, background research and analysis for the project.
- Additional Requirements and Analysis: This phase will carry out any other further planning and additional requirements thought of by the author before design.
- Design: This stage will give the author a clear direction, plan and structure for the implementation of the project.
- Coding: This stage will be the longest phase of the project, translating the code from Arduino UNO R3 to FRDM-K64F and writing code that will test the functionality and performance of both boards.
- Software Docs: Documentation regarding any useful information for the final report will be recorded here, along with all the notes concerning the process, findings and challenges of the software porting.

- Testing Code: The code will be tested to ensure it works on the robots in the correct manner.
- Software Improvements: Any faults indicated by the testing will be corrected in the code during this phase.
- Testing Robot with Boards: A comparative test between the functionality and performance of both the boards will be implemented here and the relevant data recorded.
- Project Presentation Poster: A poster will be designed during this stage for the upcoming project presentation.
- Draft Report: A draft report will be created during this stage in order to have it evaluated by and given feedback from the supervisor
- Final Report: The final report containing the process and all the results will be written during this stage.
- Project presentation: A summary of the final report will the presented at this date.

6.2 Trello

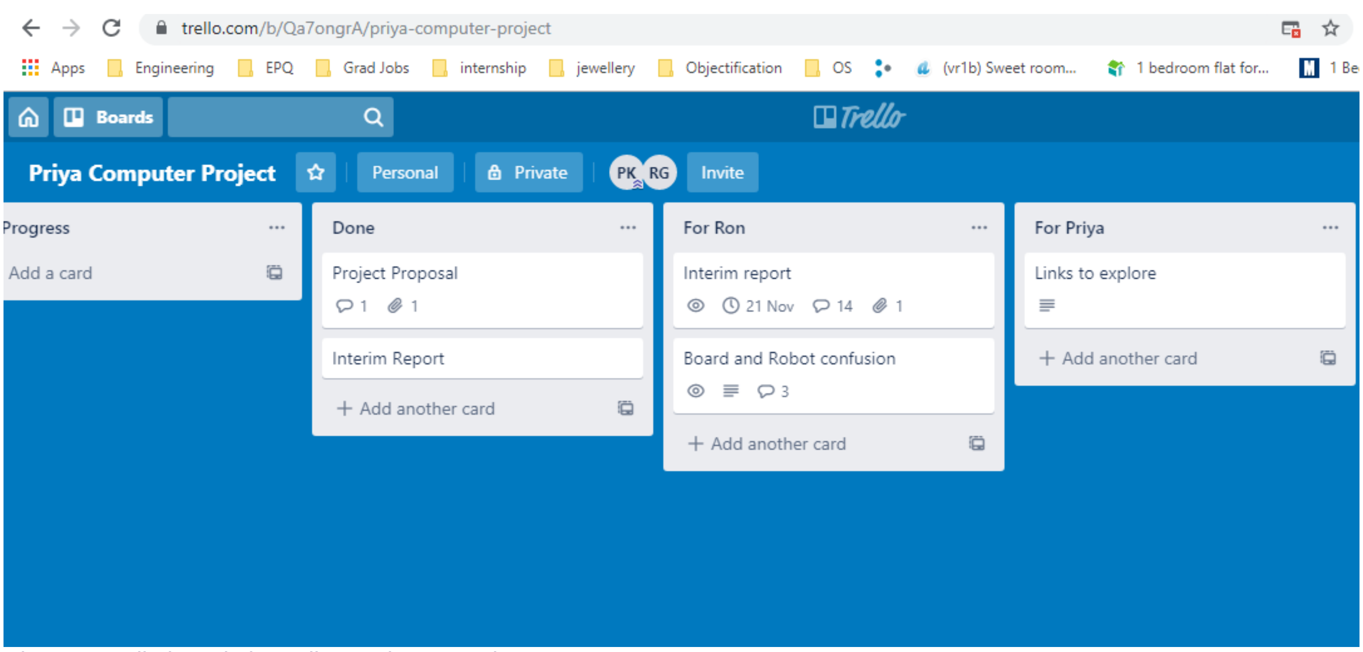


Figure 9 Trello board, the online project organiser

The author has been communicating and planning the project with their supervisor on the Trello platform and will continue to do so for the entirety of the project. It is a useful platform which consists of a board, as depicted in Figure 9, in order to organise all upcoming tasks in laid out sections and share completed work along with being able to present any questions for the supervisor or clarifications required from the supervisor. Reminders are also a feature available on this platform which aids in keeping up with deadlines.

6.2 Completed Work

At this present juncture all that has been completed is the project proposal, background research and some experimentation and familiarity with the Arduino Software (IDE) and the Mbed OS using the Zumo Robot. Some initial class diagrams have also been completed which have been presented in next section.

6.2.1. Initial Designs

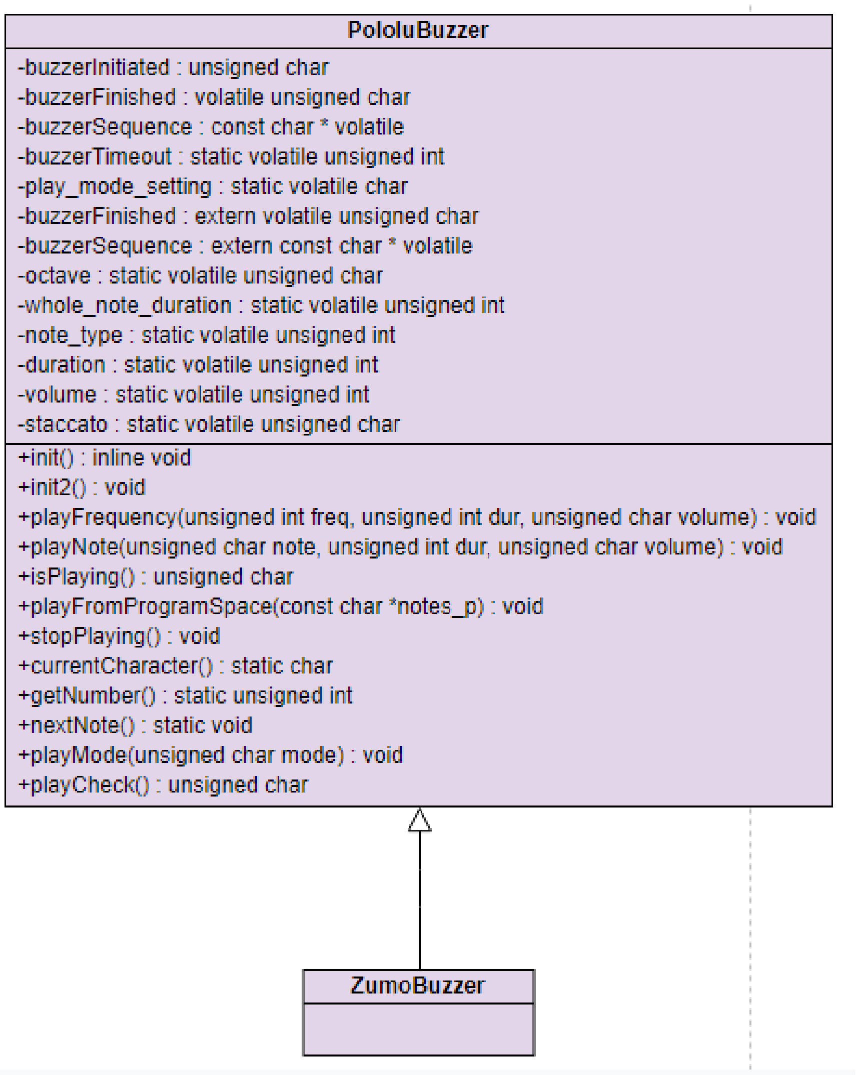
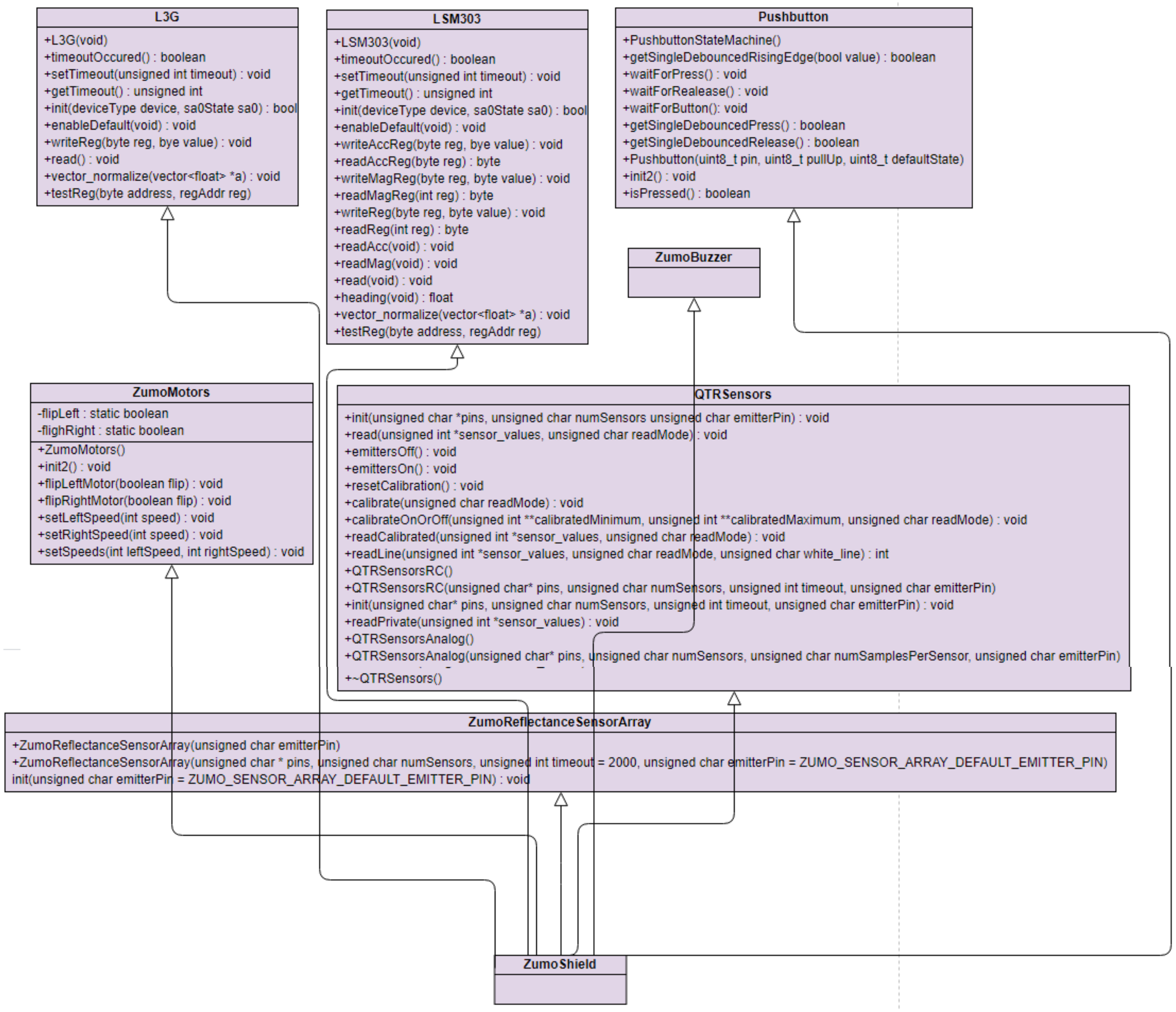

PololuBuzzer
-buzzerInitiated : unsigned char
-buzzerFinished : volatile unsigned char
-buzzerSequence : const char * volatile
-buzzerTimeout : static volatile unsigned int
-play_mode_setting : static volatile char
-buzzerFinished : extern volatile unsigned char
-buzzerSequence : extern const char * volatile
-octave : static volatile unsigned char
-whole_note_duration : static volatile unsigned int
-note_type : static volatile unsigned int
-duration : static volatile unsigned int
-volume : static volatile unsigned int
-staccato : static volatile unsigned char
+init() : inline void
+init2() : void
+playFrequency(unsigned int freq, unsigned int dur, unsigned char volume) : void
+playNote(unsigned char note, unsigned int dur, unsigned char volume) : void
+isPlaying() : unsigned char
+playFromProgramSpace(const char *notes_p) : void
+stopPlaying() : void
+currentCharacter() : static char
+getNumber() : static unsigned int
+nextNote() : static void
+playMode(unsigned char mode) : void
+playCheck() : unsigned char
ZumoBuzzer

Figure 10 UML diagram showing ZumoBuzzer as the subclass of PololuBuzzer


L3G
+L3G(void)
+timeoutOccured() : boolean
+setTimeout(unsigned int timeout) : void
+getTimeout() : unsigned int
+init(deviceType device, sa0State sa0) : bool
+enableDefault(void) : void
+writeReg(byte reg, bye value) : void
+read() : void
+vector_normalize(vector<float> *a) : void
+testReg(byte address, regAddr reg)
LSM303
+LSM303(void)
+timeoutOccured() : boolean
+setTimeout(unsigned int timeout) : void
+getTimeout() : unsigned int
+init(deviceType device, sa0State sa0) : bool
+enableDefault(void) : void
+writeAccReg(byte reg, bye value) : void
+readAccReg(byte reg) : byte
+writeMagReg(byte reg, byte value) : void
+readMagReg(int reg) : byte
+writeReg(byte reg, byte value) : void
+readReg(int reg) : byte
+readAcc(void) : void
+readMag(void) : void
+read(void) : void
+heading(void) : float
+vector_normalize(vector<float> *a) : void
+testReg(byte address, regAddr reg)
Pushbutton
+PushbuttonStateMachine()
+getSingleDebouncedRisingEdge(bool value) : boolean
+waitForPress() : void
+waitForRealease() : void
+waitForButton(): void
+getSingleDebouncedPress() : boolean
+getSingleDebouncedRelease() : boolean
+Pushbutton(uint8_t pin, uint8_t pullUp, uint8_t defaultState)
+init2() : void
+isPressed() : boolean
ZumoBuzzer
ZumoMotors
-flipLeft : static boolean
-flighRight : static boolean
+ZumoMotors()
+init2() : void
+flipLeftMotor(boolean flip) : void
+flipRightMotor(boolean flip) : void
+setLeftSpeed(int speed) : void
+setRightSpeed(int speed) : void
+setSpeeds(int leftSpeed, int rightSpeed) : void
QTRSensors
+init(unsigned char *pins, unsigned char numSensors unsigned char emitterPin) : void
+read(unsigned int *sensor_values, unsigned char readMode) : void
+emittersOff() : void
+emittersOn() : void
+resetCalibration() : void
+calibrate(unsigned char readMode) : void
+calibrateOnOrOff(unsigned int **calibratedMinimum, unsigned int **calibratedMaximum, unsigned char readMode) : void
+readCalibrated(unsigned int *sensor_values, unsigned char readMode) : void
+readLine(unsigned int *sensor_values, unsigned char readMode, unsigned char white_line) : int
+QTRSensorsRC()
+QTRSensorsRC(unsigned char* pins, unsigned char numSensors, unsigned int timeout, unsigned char emitterPin)
+init(unsigned char* pins, unsigned char numSensors, unsigned int timeout, unsigned char emitterPin) : void
+readPrivate(unsigned int *sensor_values) : void
+QTRSensorsAnalog()
+QTRSensorsAnalog(unsigned char* pins, unsigned char numSensors, unsigned char numSamplesPerSensor, unsigned char emitterPin)
+~QTRSensors()
ZumoReflectanceSensorArray
+ZumoReflectanceSensorArray(unsigned char emitterPin)
+ZumoReflectanceSensorArray(unsigned char * pins, unsigned char numSensors, unsigned int timeout = 2000, unsigned char emitterPin = ZUMO_SENSOR_ARRAY_DEFAULT_EMITTER_PIN)
init(unsigned char emitterPin = ZUMO_SENSOR_ARRAY_DEFAULT_EMITTER_PIN) : void
ZumoShield
Figure 11 UML diagram showing the inheritance of ZumoShield

References

[1] Hackster, “Arduino IoT Cloud Amazon Alexa Integration,” 2019. [Online]. Available: https://www.hackster.io/303628/arduino-iot-cloud-amazon-alexa-integration-4e6078. [Accessed 21 November 2019].

[2] Arduino, “Arduino Uno Rev3,” [Online]. Available: https://store.arduino.cc/arduino-uno-rev3. [Accessed 19 November 2019].

[3] Arm Mbed, “FRDM-K64F,” [Online]. Available: https://os.mbed.com/platforms/FRDM-K64F/. [Accessed 19 November 2019].

[4] Pololu Corporation, “Pololu Zumo Shield Arduino Library,” 2018. [Online]. Available: https://pololu.github.io/zumo-shield-arduino-library/. [Accessed 19 November 2019].

[5] J. N. Rejeev Grover, “Method of Porting Software”. USA Patent US 7,185,344 B2, 2007.

[6] s-lab, “Mbed VS Arduino,” 2013. [Online]. Available: http://slab.concordia.ca/2013/mbed/mbed-comparison-test/. [Accessed 2019 November 18].

[7] s-lab, “Mbed Overview,” 2013. [Online]. Available: http://slab.concordia.ca/2013/mbed/mbed-overview/. [Accessed 18 November 2019].

[8] J. Carver, “K64F eCompass,” 2014. [Online]. Available: https://os.mbed.com/users/JimCarver/code/K64F_eCompass/. [Accessed 19 November 2019].

[9] Mbed, “Handbook - Homepage,” [Online]. Available: https://os.mbed.com/handbook/Homepage. [Accessed 18 November 2019].

[10] Mbed, “Cookbook - Homepage,” [Online]. Available: https://os.mbed.com/cookbook/Homepage. [Accessed 18 November 2019].

[11] J. Yiu, “Technical Article - Basics of porting C-code to and between ARM CPUs: From 8-/16-Bit MCUs to Cortex-M0,” 2011. [Online]. Available: https://www.embedded.com/basics-of-porting-c-code-to-and-between-arm-cpus-from-8-16-bit-mcus-to-cortex-m0/. [Accessed 19 November 2019].

[12] Arduino, “Software,” [Online]. Available: https://www.arduino.cc/en/main/software. [Accessed 17 November 2019].

[13] Arduino, “Getting Started with Arduino UNO,” 2019. [Online]. Available: https://www.arduino.cc/en/Guide/ArduinoUno. [Accessed 17 November 2019].

[14] Arduino, “Environment,” 2015. [Online]. Available: https://www.arduino.cc/en/Guide/Environment. [Accessed 17 October 2019].

[15] Mbed, “TCP IP protocols and APIs,” 21 November 2019. [Online]. Available: https://os.mbed.com/handbook/TCP-IP-protocols-and-APIs.

[16] Mbed, “Security overview,” [Online]. Available: https://os.mbed.com/docs/mbed-os/v5.9/reference/security.html. [Accessed 21 November 2019].

[17] Arm Mbed, “Mbed 0S 5 Documentation,” [Online]. Available: https://os.mbed.com/docs/mbed-os/v5.14/reference/index.html. [Accessed 20 November 2019].

[18] B. Lim, “Moving from Arduino to Arm,” 2018. [Online]. Available: http://westsideelectronics.com/moving-from-arduino-to-arm/. [Accessed 17 November 2019].

[19] E. Styger, “The Freedom Zumo Robot,” 2013. [Online]. Available: https://mcuoneclipse.com/2013/01/31/the-freedom-zumo-robot/. [Accessed 19 November 2019].

[20] NXP, “PROCESSOR-EXPERT: Processor Expert software - Integration with CodeWarrior tool,” [Online]. Available:

https://www.nxp.com/design/software/development-software/processor-expert-software/processor-expert-software-integration-with-codewarrior-tool:PROCESSOR-EXPERT. [Accessed 19 November 2019].

[21] NXP, “Get Started with the FRDM-K64F,” [Online]. Available: https://www.nxp.com/document/guide/get-started-with-the-frdm-k64f:NGS-FRDM-K64F. [Accessed 19 November 2019].

[22] NXP, “FRDM-K64F: Freedom Development Platform for Kinetis® K64, K63, and K24 MCUs,” [Online]. Available: https://www.nxp.com/design/development-boards/freedom-development-boards/mcu-boards/freedom-development-platform-for-kinetis-k64-k63-and-k24-mcus:FRDM-K64F. [Accessed 19 November 2019].

[23] Arduino, “ATmega168/328P-Arduino Pin Mapping,” [Online]. Available: https://www.arduino.cc/en/Hacking/PinMapping168. [Accessed 18 November 2019].

[24] Atmel, “ATmega328P Datasheet,” 2015. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf. [Accessed 18 November 2019].

[25] NXP, “Kinetis K64F Sub-Family Data Sheet, Rev. 7,” 2016. [Online]. Available: https://www.nxp.com/docs/en/data-sheet/K64P144M120SF5.pdf. [Accessed 18 November 2019].

[26] E. Styger, “Zumo Line Following with FRDM-KL25Z,” 2013. [Online]. Available: https://mcuoneclipse.com/2013/02/08/zumo-line-following-with-frdm-kl25z/. [Accessed 19 November 2019].

[27] E. Styger, “Freedom Robot solves the Maze,” 2013. [Online]. Available: https://mcuoneclipse.com/2013/03/20/freedom-robot-solves-the-maze/. [Accessed 19 November 2019].

[28] L. Levin, “Getting Started with NXP FRDM-K64F and the Mbed Environment,” 2016. [Online]. Available: https://www.hackster.io/leroy2le/getting-started-with-nxp-frdm-k64f-and-the-mbed-environment-139d8d. [Accessed 19 November 2019].

[29] Pololu Corporation, “Pololu Zumo Shield for Arduino User’s Guide,” 2001-2019. [Online]. Available: https://www.pololu.com/docs/0J57/all. [Accessed 19 November 2019].

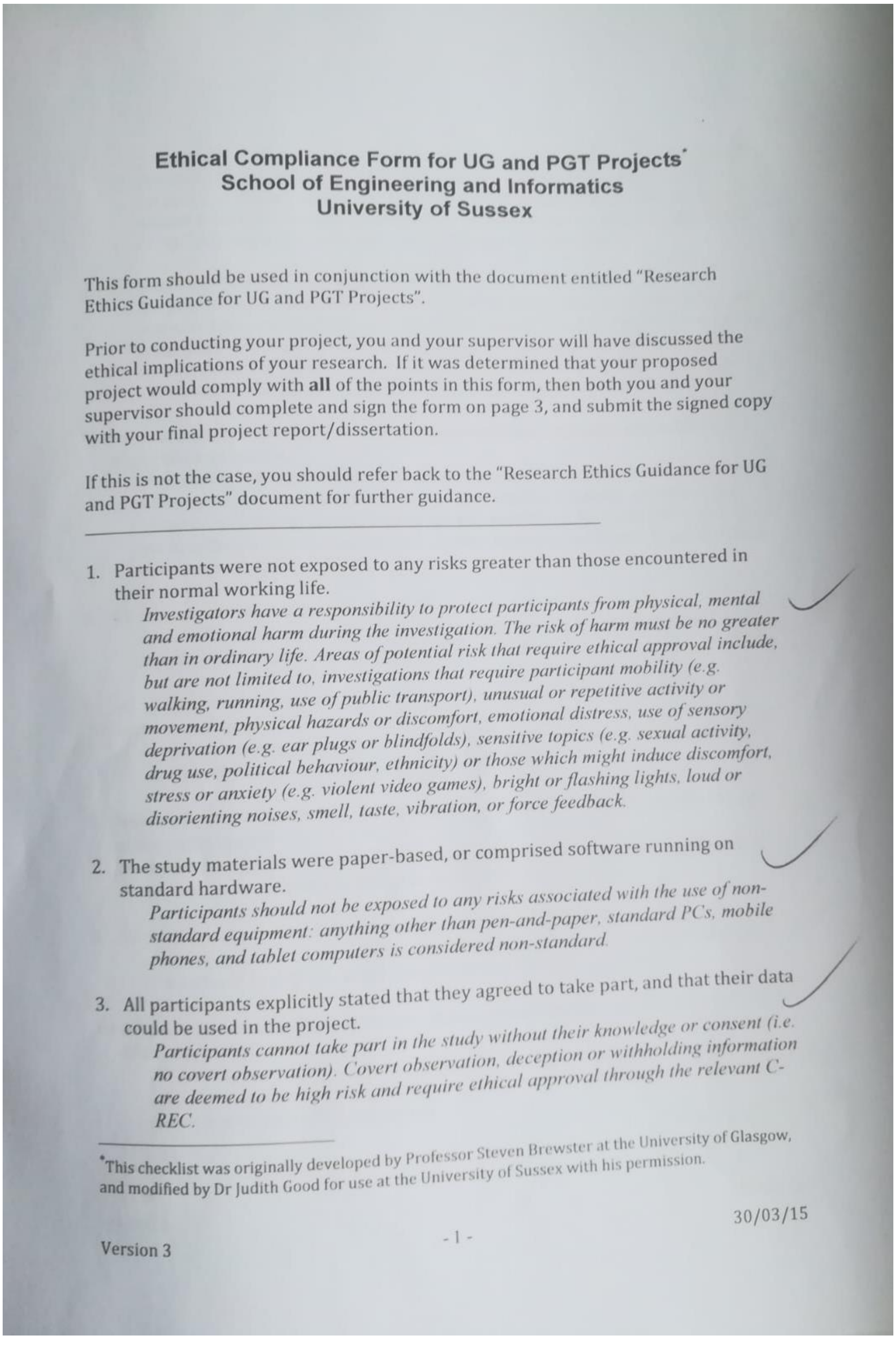
[30] Pololu, “Maze Solver,” [Online]. Available: https://www.pololu.com/docs/0J57/all#7.e. [Accessed 19 November 2019].

[31] Pololu, “Proximity Sensors and Range Finders,” [Online]. Available: https://www.pololu.com/category/189/proximity-sensors-and-range-finders. [Accessed 19 November 2019].

Appendix

Interim Log

Date	Meeting Contents
7/10/19	Overview of deadlines, useful resources, useful beginner practise tasks to familiarise oneself with ZUMO, relevant suggestions, Trello platform
13/11/19	Reminder of essential requirements for Interim report, signing the ethical compliance form



Ethical Compliance Form for UG and PGT Projects[*]
School of Engineering and Informatics
University of Sussex

This form should be used in conjunction with the document entitled "Research Ethics Guidance for UG and PGT Projects".

Prior to conducting your project, you and your supervisor will have discussed the ethical implications of your research. If it was determined that your proposed project would comply with **all** of the points in this form, then both you and your supervisor should complete and sign the form on page 3, and submit the signed copy with your final project report/dissertation.

If this is not the case, you should refer back to the "Research Ethics Guidance for UG and PGT Projects" document for further guidance.

---

1. Participants were not exposed to any risks greater than those encountered in their normal working life. ✓
   Investigators have a responsibility to protect participants from physical, mental and emotional harm during the investigation. The risk of harm must be no greater than in ordinary life. Areas of potential risk that require ethical approval include, but are not limited to, investigations that require participant mobility (e.g. walking, running, use of public transport), unusual or repetitive activity or movement, physical hazards or discomfort, emotional distress, use of sensory deprivation (e.g. ear plugs or blindfolds), sensitive topics (e.g. sexual activity, drug use, political behaviour, ethnicity) or those which might induce discomfort, stress or anxiety (e.g. violent video games), bright or flashing lights, loud or disorienting noises, smell, taste, vibration, or force feedback.

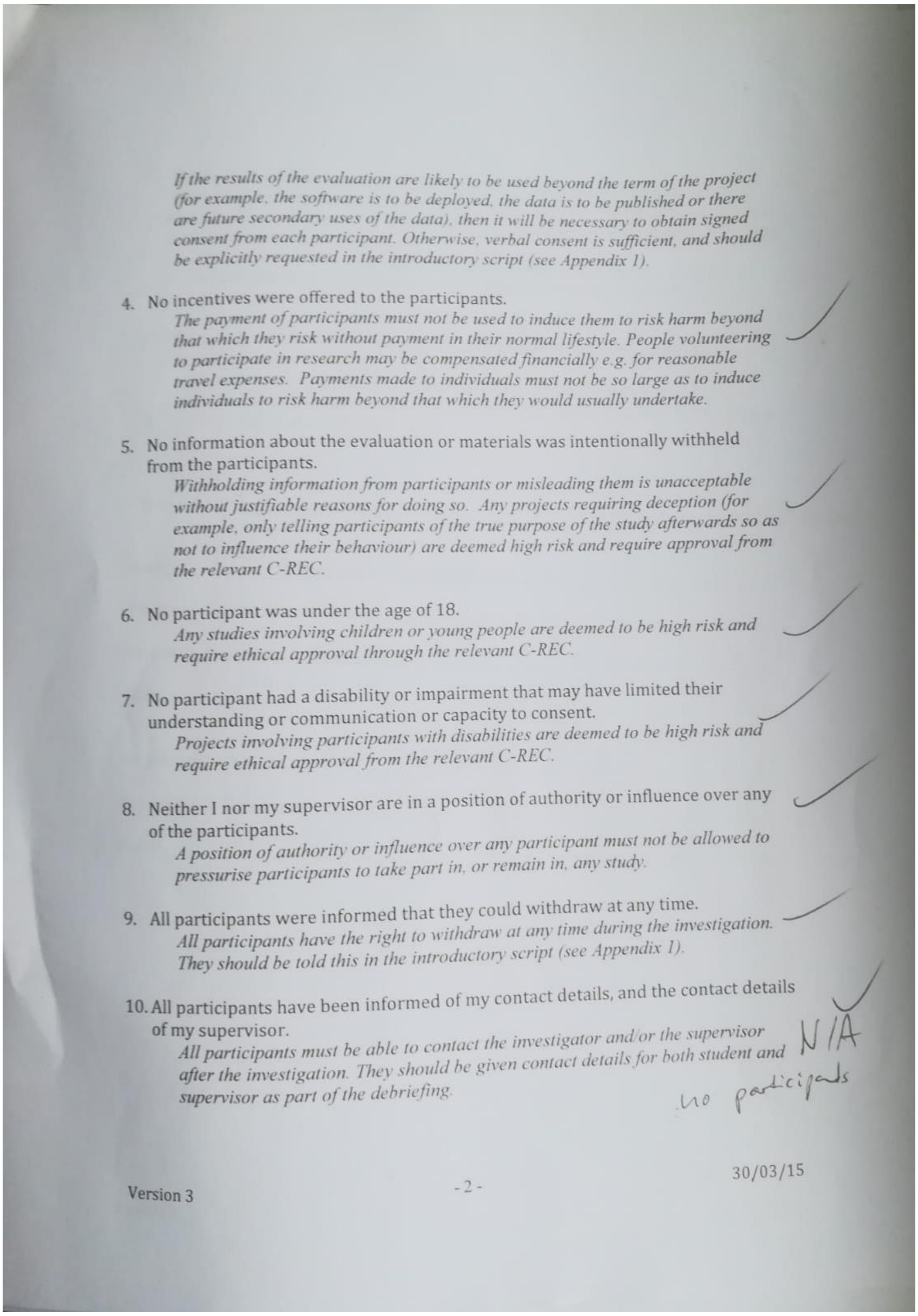
2. The study materials were paper-based, or comprised software running on standard hardware. ✓
   Participants should not be exposed to any risks associated with the use of non-standard equipment: anything other than pen-and-paper, standard PCs, mobile phones, and tablet computers is considered non-standard.

3. All participants explicitly stated that they agreed to take part, and that their data could be used in the project. ✓
   Participants cannot take part in the study without their knowledge or consent (i.e. no covert observation). Covert observation, deception or withholding information are deemed to be high risk and require ethical approval through the relevant C-REC.

[*] This checklist was originally developed by Professor Steven Brewster at the University of Glasgow, and modified by Dr Judith Good for use at the University of Sussex with his permission.

30/03/15

Version 3 - 1 -



If the results of the evaluation are likely to be used beyond the term of the project (for example, the software is to be deployed, the data is to be published or there are future secondary uses of the data), then it will be necessary to obtain signed consent from each participant. Otherwise, verbal consent is sufficient, and should be explicitly requested in the introductory script (see Appendix 1).

4. No incentives were offered to the participants. ✓
   The payment of participants must not be used to induce them to risk harm beyond that which they risk without payment in their normal lifestyle. People volunteering to participate in research may be compensated financially e.g. for reasonable travel expenses. Payments made to individuals must not be so large as to induce individuals to risk harm beyond that which they would usually undertake.

5. No information about the evaluation or materials was intentionally withheld from the participants. ✓
   Withholding information from participants or misleading them is unacceptable without justifiable reasons for doing so. Any projects requiring deception (for example, only telling participants of the true purpose of the study afterwards so as not to influence their behaviour) are deemed high risk and require approval from the relevant C-REC.

6. No participant was under the age of 18. ✓
   Any studies involving children or young people are deemed to be high risk and require ethical approval through the relevant C-REC.

7. No participant had a disability or impairment that may have limited their understanding or communication or capacity to consent. ✓
   Projects involving participants with disabilities are deemed to be high risk and require ethical approval from the relevant C-REC.

8. Neither I nor my supervisor are in a position of authority or influence over any of the participants. ✓
   A position of authority or influence over any participant must not be allowed to pressurise participants to take part in, or remain in, any study.

9. All participants were informed that they could withdraw at any time. ✓
   All participants have the right to withdraw at any time during the investigation. They should be told this in the introductory script (see Appendix 1).

10. All participants have been informed of my contact details, and the contact details of my supervisor. ✓ N/A
    All participants must be able to contact the investigator and/or the supervisor after the investigation. They should be given contact details for both student and supervisor as part of the debriefing.
    no participants

Version 3 - 2 - 30/03/15

Proposal: Robot Control Interface – From Arduino to ARM Mbed

The ZUMO robot for the Arduino platform is well-established, with plenty of libraries available within its software. However, while ports have been created for certain ARM Mbed platforms (like the K25), there is currently none specifically for the K64F. The purpose of this project will be to remove the Arduino ATmega32U4 and translate its library codes (ZumoShield) [32] to create a C++ library that runs on the more powerful ARM K64F controller. It can then be used to control the ZUMO robot and its sensors and actuators (motors, infrared, etc.) and the difference in the robot's performance on the new platform will be evaluated.

The Arduino and K64F controllers are both programmed in C++ so the specific translation of code is quite achievable due to the similarities in the languages. The main focus on the programming end will be to explore the concept of "software porting"; investigating the process of adapting software for use on an alternate platform to the one it was originally designed to execute on. I.e. altering software for it to be usable in different environments. This means one should firstly consider the design methods best supported by both languages in particular — C follows a top-down approach while C++ follows bottom-up programming. The paradigms for C and C++ are imperative (structural) and object-oriented respectively, C supporting a structure with more computational steps and C++ focusing on hierarchies. More design is required in the development of an imperative language, therefore translating to C++ will be easier than the reverse. The benefit of the use of objects will also make the library better organised, taking advantage of the additional features of C++. The most essential aim is to be aware of the native capabilities of both languages in order to focus on the best way to execute the desired functionality instead of focusing on the manner in which it was originally coded.

Additionally, one must consider the key differences in the programming environment for the two microcontrollers as their structure is developed to support the unique functionality of both boards. These environments are provided by Arduino and ARM Mbed, namely the Arduino Software (IDE) supporting any Arduino board [33] and the Mbed OS doing the same for Mbed boards. Once the board is specified both IDE's include built in support for them, Arduino using the "Board Manager". This board manager "sets the parameters (e.g. CPU speed and baud rate) used when compiling and uploading sketches; and sets the file and fuse settings used by the burn bootloader command" [34]. The K64F is "fully supported in the mbed platform, so it gets access to the free tools and SDK that provides experienced embedded developers with powerful and productive tools for building proof-of-concepts" [5]. The Mbed OS provides its own APIs which allow code to work on different microcontrollers in a uniform way. This reduces a lot of the challenges in getting started with microcontrollers [35].

The hardware itself must also be compared, primarily the pin configurations which differ marginally between the microcontrollers. The pinout diagrams show the layout of the pins (which make up ports); the Arduino UNO R3 containing 28 pins and FRDM-K64F containing 64. The diagrams illustrate "…the mapping between Arduino pins and ATmega328P ports". "The freedom board headers enable up to 64-pins and give access to most of the Kinetis K64F signals. Outer row pins deliver right signals to meet Arduino R3 standard, the inner row is connected to up to 32 additional Kinetis K64F pins". The pinout of the Kinetis K64F MCU show the signal connections with the board components (RGB LED, Motion Sensors) and extension sensors (SD Card, Bluetooth, WiFi). The pins on each board are laid out in a similar manner particularly with the 32 outer pins of the FRDM-K64F; which will greatly assist in understanding the parallels between the pin signals in order to match and adapt functionality from one board's pins to another: easing the process of software porting. The crucial distinction in regards to the hardware is the additional functionality provided by the 32 extra inner pins of the FRDM-K64F. That being said, this will not have any implications on the software porting itself, but more on other requirements set out in this report such as testing the difference in the functionality of the Zumo robot between the two boards by writing further code that will utilise the potential extra functionality from the additional pins.

Primary Objectives

The above describes the in-depth general purpose of the project, all of which are guaranteed to be achieved as they will remain the primary overall focus. However, a more detailed list of examples of investigation will be listed below.

Concepts evaluated when comparing boards:

- Timing including code run time & response time
- Functionality – the variety of useful functions that can be implemented on each board
    - ⇒ K64F 6-axis combo Sensor Accelerometer and Magnetometer [28]
- Quality of performance i.e. error-level e.g. driving 10cm, accuracy of each board
- Number of steps taken to execute code
- Comparing the quality of execution of the sensors that come with ZUMO incl.
    - ⇒ Quadrative and inertial encoders (accelerometer & gryo – motor and robot following straight line without interference)
    - ⇒ Reflectance sensors (infrared LED sensors e.g. to test how accurately it can follow a straight line)

Extensions

My extension would explore additional sensors that can be purchased for less than $5 and comparing the performance of both boards with these sensors:

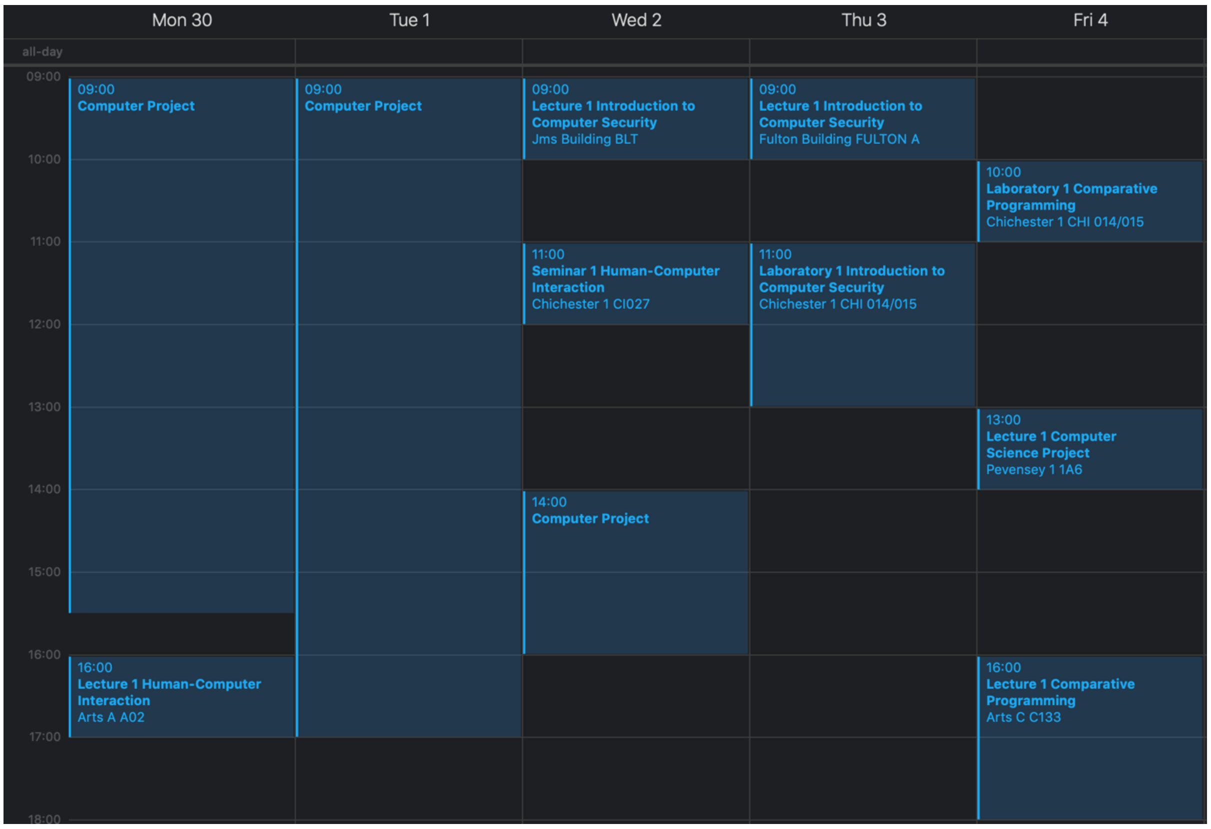
- Sensors for ZUMO e.g. optical or sonar range finders
    - ⇒ (the Zumo 32U4 already has built-in IR proximity sensors, but additional sensors can be incorporated for increased range or detection area) [8]
- Sensors that can be attached to the Arduino and K64F boards incl: proximity sensors, eCompass, temperature & humidity sensor, light sensors, collision sensors [6][2]
- Other added modules incl. vibration, joystick, digit display [6][1]

Observing if the boards differ in performance when combining sensors with functionality.

This project relates to my degree course as it utilises programming skills and knowledge along with providing a better understanding of lower-level communication with machines and hardware that may be useful when combining computer science with Engineering. The only resources required are the ZUMO robot and two MCUs which have been provided by the department (any extension resources will be purchased by myself if necessary).

Date	Meeting Contents
7/10/19	Overview of deadlines, useful resources, useful beginner practise tasks to familiarise oneself with ZUMO, relevant suggestions, Trello platform

[2] Sensors provided by SEEED studios



Mon 30	Tue 1	Wed 2	Thu 3	Fri 4
09:00 Computer Project	09:00 Computer Project	09:00 Lecture 1 Introduction to Computer Security Jms Building BLT	09:00 Lecture 1 Introduction to Computer Security Fulton Building FULTON A	10:00 Laboratory 1 Comparative Programming Chichester 1 CHI 014/015
16:00 Lecture 1 Human-Computer Interaction Arts A A02		11:00 Seminar 1 Human-Computer Interaction Chichester 1 CI027	11:00 Laboratory 1 Introduction to Computer Security Chichester 1 CHI 014/015	13:00 Lecture 1 Computer Science Project Pevensey 1 1A6
		14:00 Computer Project		16:00 Lecture 1 Comparative Programming Arts C C133

Figure 12Personal weekly timetable indicating lectures and periods of the week intended to devote to the project.

References

[1] Pololu, "Zumo Shield Arduino Library," [Online]. Available: https://github.com/pololu/zumo-shield-arduino-library.

[2] Arduino, "Software," [Online]. Available: https://www.arduino.cc/en/main/software.

[3] Arduino, "Environment," [Online]. Available: https://www.arduino.cc/en/Guide/Environment.

[4] Arm Mbed, "Overview - Reference | Mbed 0S 5 Documentation," [Online]. Available: https://os.mbed.com/docs/mbed-os/v5.14/reference/index.html.

[5] L. Levin, "Getting Started with NXP FRDM-K64F and the Mbed Environment," 2016. [Online]. Available: https://www.hackster.io/leroy2le/getting-started-with-nxp-frdm-k64f-and-the-mbed-environment-139d8d. [Accessed 19 November 2019].