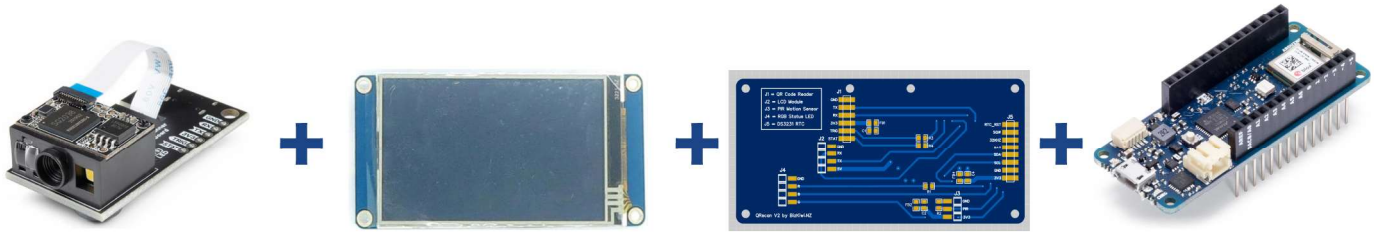
QR Code Scanning Kiosk

Development of a QR Code Scanning Kiosk for use in a Retail Customer Loyalty Programme.



MG7101 Engineering Development Project

Final Report

Bachelor of Engineering Technology (Electrical)

Department of Electrotechnology

Unitec Institute of Technology

Friday 26 November 2021

Stephen Julian

Student ID: 1041664

Project Supervisor

Morgan Look

Declaration of Originality

This report is my own unaided work and was not copied from nor written in collaboration with any other person.

Name: Stephen Julian

Date: Tuesday 23/11/2021

Abstract

This report is about the development of a device for scanning barcodes and QR codes as part of a customer loyalty programme enabling customers to collect rewards from retailers. The system may be of interest to quick service restaurants, cafés and the hospitality industry as well as retailers in general.

There are eight parts to this report:

1. Introduction
2. Background
3. Research Scope
4. Methodology
5. Results
6. Manufacturing Costs
7. Timeline
8. Conclusions

Part 1 introduces the project overview and the motivation behind the efforts.

Part 2 discusses types of existing scanning systems and how they fit in the marketplace.

Part 3 details the research scope and outlines a vision for a new QR code scanning system with particular focus on an application as a customer loyalty scheme for a quick service restaurant.

Part 4 introduces the methodology as the building of a series of rapid prototypes with each one building upon the outcomes and lessons learned from the previous ones.

Part 5 documents the development of the prototypes in detail.

Part 6 covers the cost of materials and manufacturing of the series of prototypes.

Part 7 references timelines of the project.

Part 8 summarises the outcomes of the project and includes recommendations for improvements in future development work.

Acknowledgements

Special thanks for helping make this project possible go to:

- Momen Bahadornejad, my project tutor.
- Morgan Look, my project supervisor and manufacturing tutor.
- Dr. Nigel Yee, my microcontrollers and embedded systems tutor.

Table of Contents

1. Introduction

1.1 Terms of Reference

This report documents the work of student Stephen Julian with the MG7101 Engineering Development Project Semester A paper at Unitec Institute of Technology. This work has taken place under the tuition of Momen Bahadornejad and the supervision of tutor Morgan Look.

1.2 Procedure

Information and content for this report was gathered from photographs, diagrams and screenshots taken during the project as well as some content sourced from Internet websites.

1.3 Project Overview

The purpose of this project was to build prototype versions of an electronic device designed for use by vendors in the restaurants, cafés and hospitality industry to connect customers' loyalty cards and/or mobile apps with the digital infrastructure of the vendor. Use cases may extend to include many types of retail stores in general.

1.4 Motivation

Use cases and applications of the proposed device and potential variants include:

- Customer loyalty programmes
- Identity verification and access control
- Retail payment and promotion services

1.5 Market Research

Devices that scan barcodes, QR codes and/or NFC tags. Some devices enable the user to use their smartphone to scan barcodes, QR codes or NFC tags. Mobile apps are available that add this functionality to the smartphone in a user-centric and vendor controlled way.

2. Background (Literature Review)

2.1 Existing Systems

2.1.1 Components, Functions and Features

Customer loyalty programme membership cards may feature magnetic strips, barcodes, Quick Reaction (QR) codes or Near Field Communication (NFC) tags. These are all different mechanisms of encoding a unique identification token for recognising or looking up a unique identifier in a database to enable the customer to collect rewards points or receive a reward for having done so.

Broadly, the market splits into 3 categories with devices forwarding data via:

- Wireless Bluetooth (BT) and/or Bluetooth Low Energy (BLE). These devices tend to be handheld or standalone scanning devices.
- Universal Serial Bus (USB) wired cable connection. These devices tend to be countertop devices fixed in place.
- WiFi or wired ethernet LAN connection. Ethernet devices are fixed in place.

Some devices include an LCD Display to provide the customer with instructions for use and providing feedback about the result and the status of their rewards and balance of points.

3. Research Scope

3.1 Prototypes Overview

3.1.1 Components, Functions and Features

Specific prototypes will have components, functions and features as follows.

Identification (ID) and information scanning or reading:

- 1D Barcode (including Universal Product Codes (UPC))
- 2D Barcode and Quick Response (QR) Codes

Data communications connectivity with:

- A tablet, computer or smartphone paired with the scanner via BT or BLE radio.
- An internet server via WiFi connectivity with Local Area Network (LAN).
- Universal Serial Device (USB) connectivity with any PC supporting USB 2.0 host or later.

Human Interface Device (HID) sensors and actuators:

- Liquid Crystal Display (LCD) to display instructions / visual feedback to user.
- Touchscreen Reader for the user to interact with the device.
- Light Dependent Resistor (LDR) for detecting a nearby device screen.
- Passive Infrared Device (PIR) for detecting proximity of a person.

3.1.2 Plastic Membership Card

A plastic membership card helps customers to easily download a mobile app.

The card features:

- a printed QR code.
- an embedded NFC tag.
- both a QR code and an NFC tag contain the same encoded website link.
- The website has marketing material for the loyalty programme.
- The website links to app download pages on Google Play, Apple App Store.

3.1.3 Mobile Application

The free mobile app enables the customer to:

- collect order receipts.
- submit customer feedback.
- collect rewards points.
- redeem rewards points scanning QR code in the app on the scanning device.
- surrender rewards points in exchange for a donation to a charity.

3.1.4 Scanning Device

The scanning device:

- reads the QR code scanned by the device.
- reads a unique customer ID from the QR code
- sends customer ID to the tablet, computer or smartphone of the store.
- receives event response status back.
- displays response status information to the customer on the scanner LCD.

3.1.5 Store Tablet, Computer or Smartphone

The tablet, computer or smartphone device belonging to the store:

- receives the Customer ID from the scanning device.
- requests available Rewards Points for Customer from a Cloud based server.
- displays available Rewards Points for a Customer to a store employee.
- sends response status back to the connected scanner to display on the LCD.

3.1.6 Cloud Web Server

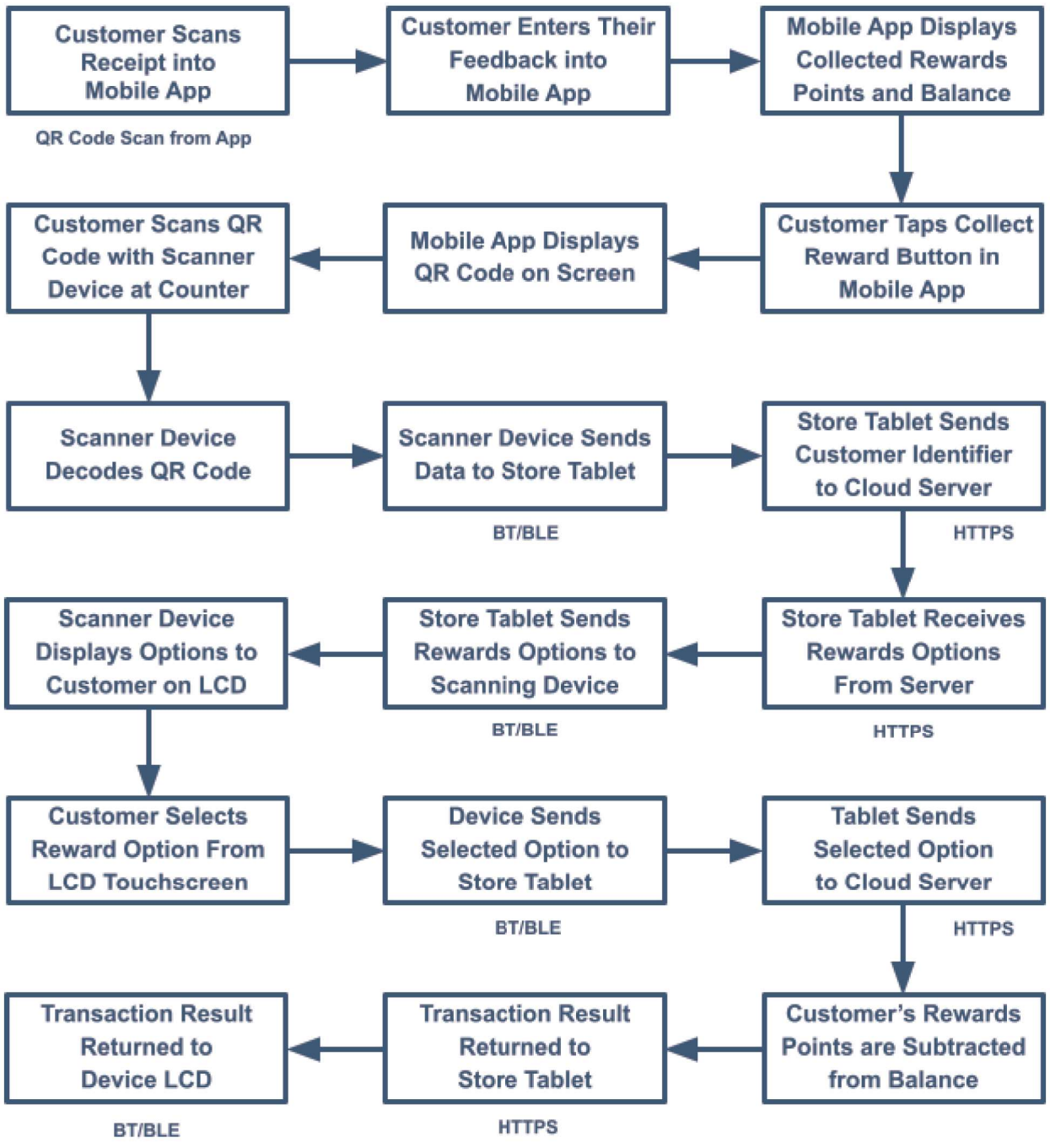
The cloud based web server is a secure hosting infrastructure that:

- stores Customer and Rewards Points information.
- is updated by the mobile app on the customer's smartphone.
- is checked by the tablet, computer or smartphone belonging to the store.
- adds Rewards Points when a customer submits feedback on an order.
- removes Points when they are exchanged for rewards or a charity donation.
- generates Reports showing points, exchanges and donations activity.
- informs management staff of charity donations deposit amounts required.

3.1.7 Prototypes Scope

Project scope for the practical work is limited to the Scanning Device prototypes.

3.2 Block Diagram of Proposed System



Block Diagram of Proposed System (Chart I)

3.3 Customer Receipts with QR Codes

3.3.1 Customer Receipts in a Loyalty Programme

By encoding order information into a QR code and including this on the receipt a customer is then able to scan the QR code into their smartphone using a loyalty programme app. This provides a fast and easy method for customers to include relevant and useful information when participating in a customer loyalty programme.

Such information typically found included on a customer receipt includes:

- Date and time of transaction.
- Unique store, transaction, register, employee and supervisor identifiers.
- Products and services that were purchased including the quantities.
- Unit prices, total amount, payment method, amount tendered, change given.

3.3.2 Encoding Order Information in a QR Code

Open source code libraries are available designed to generate a QR code from parameters passed in. When scanned, a generated QR code can be decoded into a structured data set e.g. in JSON or XML formats. The structured data can then be processed and manipulated or stored. For the purpose of testing the prototypes QR codes can be automatically generated by free online web services. The following example shows the text "Order Number 747" encoded into a QR code that is now ready for scanning.

3.3.3 Customer Receipt with a QR Code

Your order number is

125

MkDonnell's Queenstown
Tel: (03) 442 9796
GST: 996 232 982

TAX INVOICE

ORD #25 REG #1 08/07/2021 14:28:49

QTY ITEM	TOTAL
1 Large Choc Shake	5.00
SubTotal	5.00
Take-Out Total (incl GST)	5.00
EFTPOS	5.00
TOTAL INCLUDES GST OF	0.65

------EFTPOS------
TERMINAL 94003340
TIME 08JUL21 14:30
TRAN 008056 CREDIT
MCARD
CARD 4164
CONTACTLESS
Debit Mastercard
RID: A000000004
PIX: 1010
ARQC:
BFA540D671A59E53
TVR: 0000008001
ATC: 0238
TSI: 0000
AUTH F89945
PURCHASE NZ$5.00
TOTAL NZ$5.00
ACCEPTED

CUSTOMER COPY

Scan this Code to collect Rewards Points

WANT FREE FOOD?
Download the Rewards App, scan QR codes to collect points and exchange them for free food!

Get the App at mkdonnells.com

Customer Receipts from a Quick Service Restaurant (Figure I a, b)
a: Typical receipt concept from a Quick Service Restaurant.
b: Concept composite image showing how a customer receipt with QR code may appear.

3.3.4 Receipt QR Code Scanning Process

The rewards mobile app scans a QR code on the receipt containing this URL: https://mkdonnells.com/?qrcodeid=[validuniqueid]&countryid=nz&storeid=63&regid=1&orderid=21&date=2021-07-08&time=14:28:49&ordervalue=5.00&numberpoints=[encryptedtoken]

If the QR code is scanned via a third-party mobile app then the scanned link will not connect to the rewards points server. This is because for security reasons the QR code excludes both the rewards points server address and the required mobile app customer ID.

The official rewards points mobile app replaces the chars "https://mkdonnells.com/" with "https://mkserver.com/rewards/points/" in the URL string. This enables the app to connect to the correct secure server that tallies rewards points for customers and the correct page on the server.

The mobile app will also append the customer ID to the scanned URL. Additional data may also optionally automatically be appended for collection by the server. This is dependent on the business logic in the mobile app and this can be updated as new versions of the app are released. For example, additional data collected may include the datetime the QR code is scanned (not the datetime of the purchase). However, the pixel density of the QR code is proportional to the quantity of data being encoded and higher pixel densities can lead to slow scanning times due to increased error rates.

The server logs the GET request and inserts the relevant data in a database table and updates the balance of rewards points for the account of the submitted mobile app user ID. The QR code ID is in the form of a long, unique, unsequenced and pseudorandom string of chars. Once used, the record for the QR code ID in the database table is flagged so that it cannot be used again to collect rewards points.

After processing the request the server returns a transaction result string to the mobile app in the form of a JSON string containing data parameters. For example:

```
{
        result="success",
        pointsbalance=5;
}
```

In a basic system the mobile app may send a GET request to the server at the URL: https://mkserver.com/rewards/points/?qrcodeid=[validuniqueid]&countryid=nz&storeid=63&regid=1&orderid=21&date=2021-07-08&time=14:28:49&ordervalue=5.00&numberpoints=[encryptedtoken]&custid=[validappuserid]

3.3.5 Receipt QR Code Data Parameters

Name	Example Value	Description, Format
qrcodeid		[unique, pseudorandom encrypted value]
countrycode	NZ	[2 letter country code]
regionid	Queenstown	[string]
storeid	63	[integer]
regid	1	[integer]
orderid	21	[integer]
orderdate	2021-07-08	[string]
ordertime	14:28:49	[string]
orderitems		[encrypted delimited String]
ordervalue		[encrypted value]
rewardpoints		[encrypted value]
custid		[encrypted app user id]

Receipt QR Code Data Parameters (Table 1)

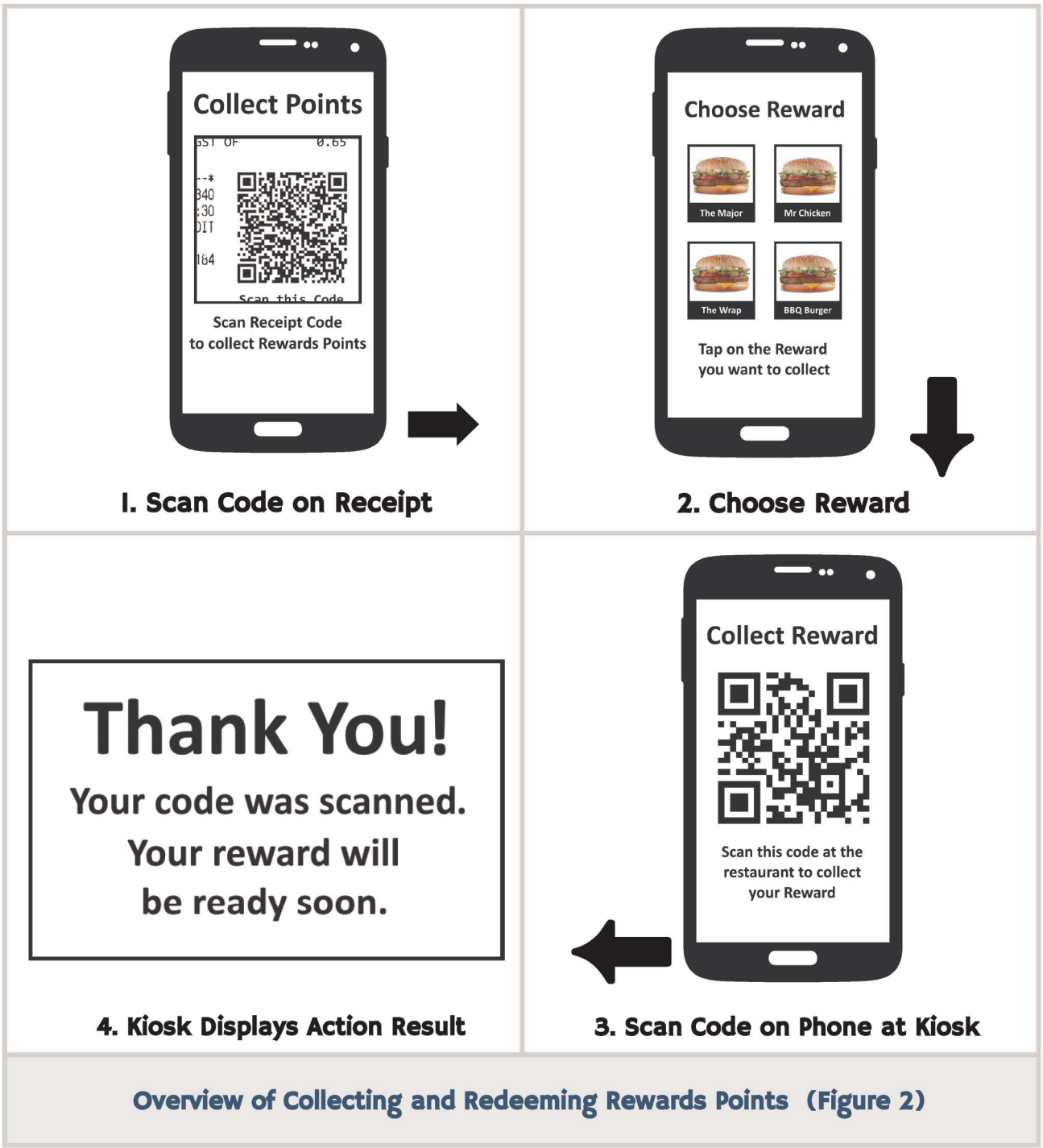
3.3.6 Summary of Receipt Scanning

A consumer oriented server address could instead be included in the QR code URL that will guide the user to download the official mobile app required for collecting and redeeming rewards points.

The above thus outlines how rewards points collecting could work with the Mobile App and server systems.

Important note: To improve security in a production system the parameters passed in the URL string via the GET request should instead be sent to the server as parameters in a POST request to prevent potential interception of the data enroute to the server.

3.4 Collecting and Redeeming Rewards Points



Overview of Collecting and Redeeming Rewards Points (Figure 2)

4. Methodology

4.1 Series of Rapid Progressive Prototypes

A series of rapid incremental prototypes will be developed, each with features and functionality improving on the previous prototype.

New in Prototype V1	New in Prototype V2	New in Prototype V3	New in Prototype V4	New in Prototype V5
Arduino Nano IoT 33 MCU	**Arduino Nano IoT 33 MCU**	**Arduino MKR WiFi 1010 MCU**	**Arduino MKR WiFi 1010 MCU**	**Arduino MKR WiFi 1010 MCU**
Wired Breadboard Circuit	**Custom Adapter PCB V1**	**Custom Adapter PCB V2**	**Custom Adapter PCB V2**	**Custom Adapter PCB V3, Rev.2**
Barcode and QR code scanning	Modules interfaced via headers	3.7V LiPo Rechargeable Battery	ArduinoBearSSL TLS and SSL	User Guide, Status on LCD
Device status notification	Components Soldered to PCB	WiFi Internet Connectivity	MQTT Messaging with AWS IoT	LCD Touchscreen Interaction
Human proximity detection	Decoupling Capacitors.	Accurate Time and Date	AWS IoT Core	Device Enclosure Designed
Prototype circuit connectivity	Ferrite Bead Filters.	DS3231 Real-Time Clock (RTC)	AWS DynamoDB	Device Enclosure 3D Printed
Current limiting to protect pins		NTP Time Server Sync to RTC		Device fully assembled
Pulldown Resistors				DHT-12 Thermistor
				Piezoelectric Buzzer SMD
Functionality Tested	**Reliability Tested**	**Usability Tested**	**Connectivity Tested**	**Practicality Tested**

Series of Rapid Progressive Progressive Prototypes (Table 2)

5. Results

5.1 Prototype V1 - Breadboard System

5.1.1 Prototype V1 Goals

The objective for Prototype V1 was to complete a simple functional prototype and successfully test this to ensure the components and system operate as expected. This first prototype would be in the form of a wired breadboard circuit featuring most of the modules and components intended for use in the project.

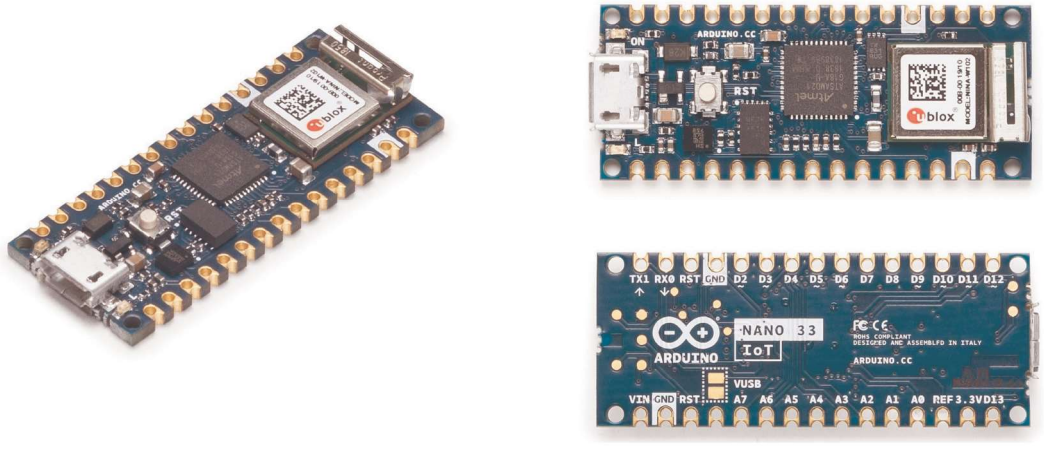
Objectives for Prototype V1 are to design, implement and test a breadboard circuit:

1. Arduino Nano IoT 33 MCU
2. QR Code scanning module
3. PIR motion sensor
4. RGB LED status indicator
5. Test sensors and actuators (including PIR motion sensor and RGB LED)

With this prototype the Arduino will emulate a keyboard when interfaced with the Tablet via USB connection. The Tablet will function as a USB host.

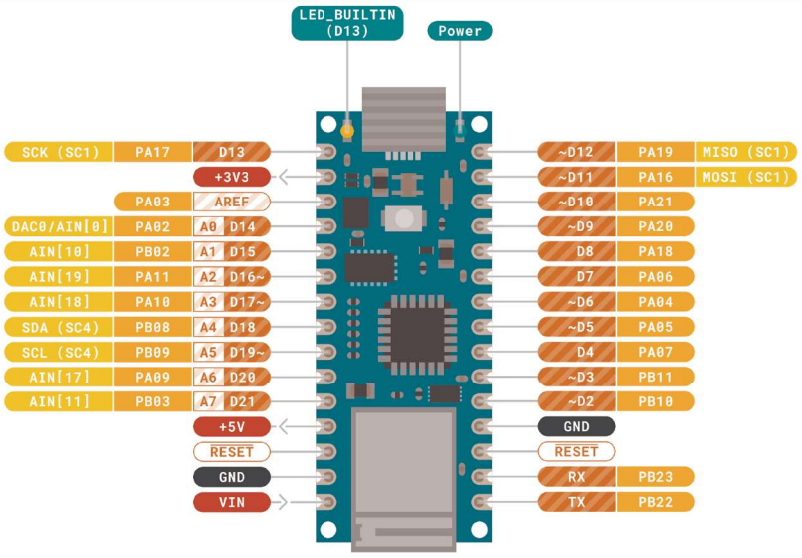
5.1.2 Arduino Nano 33 IoT Microcontroller

The Arduino Nano 33 IoT is a significant upgrade to earlier Arduino Nano microcontrollers. Key new features include a WiFi and BLE wireless communication module, hardware cryptography, 3.3V architecture, configurability for supporting 3 serial ports and castellated edges allowing soldering flat on to a breakout PCB.



Arduino Nano 33 IoT Microcontroller (Figure 3 a, b, c)

5.1.3 Arduino Nano 33 IoT Microcontroller Pinout



Arduino Nano 33 IoT Microcontroller Pinout (Figure 4)

5.1.4 Arduino Nano 33 IoT Specifications

Feature / Function	Specification
Microcontroller	SAMD21 Cortex-M0+ 32bit low power ARM MCU
Radio Module	U-blox NINA-W102
Secure Element	ATECC608A
Clock	48 MHz
Memory	CPU Flash Memory: 256KB, SRAM: 32KB, EEPROM: None
Connectivity	WiFi, BLE 4.2
Digital Input / Output Pins	14
PWM Pins	11 (2, 3, 5, 6, 9, 10, 11, 12, 16 / A2, 17 / A3, 19 / A5)
Interfaces	USB, SPI, I2C, I2S, UART
Analog Input Pins	8 (ADC 8/10/12 bit)
Analog Output Pins	1 (DAC 10 bit)
External Interrupts	All digital pins
LED_BUILTIN	13
USB	Native in the SAMD21 Processor
IMU	LSM6DS3
Voltages	Input USB: 5V, Input VIN: 4.5-21V, Operating: 3.3V
Currents	DC per IO Pin Limit: 7mA
Pinout	Digital: 14, PWM: 6, Analog: 8
Dimensions	18 x 45 mm (5 gm with headers)

Arduino Nano 33 IoT - Technical Specifications (Table 3)

5.1.5 Voltage Limits on the Arduino Nano 33 IoT

"The microcontroller on the Arduino Nano 33 IoT runs at 3.3V, which means that you must never apply more than 3.3V to its Digital and Analog pins. Care must be taken when connecting sensors and actuators to assure that this limit of 3.3V is never exceeded. Connecting higher voltage signals, like the 5V commonly used with the other Arduino boards, will damage the Arduino Nano 33 IoT.

"The microcontroller on the Arduino Nano 33 IoT runs at 3.3V, which means that you must never apply more than 3.3V to its Digital and Analog pins. Care must be taken when connecting sensors and actuators to assure that this limit of 3.3V is never exceeded. Connecting higher voltage signals, like the 5V commonly used with the other Arduino boards, will damage the Arduino Nano 33 IoT.

To avoid such risk with existing projects, where you should be able to pull out a Nano and replace it with the new Nano 33 IoT, we have the 5V pin on the header, positioned between RST and A7 that is not connected as default factory setting. This means that if you have a design that takes 5V from that pin, it won't work immediately, as a precaution we put in place to draw your attention to the 3.3V compliance on digital and analog inputs.

5V on that pin is available only when two conditions are met: you make a solder bridge on the two pads marked as VUSB and you power the Nano 33 IoT through the USB port. If you power the board from the VIN pin, you won't get any regulated 5V and therefore even if you do the solder bridge, nothing will come out of that 5V pin. The 3.3V, on the other hand, is always available and supports enough current to drive your sensors. Please make your designs so that sensors and actuators are driven with 3.3V and work with 3.3V digital IO levels. 5V is now an option for many modules and 3.3V is becoming the standard voltage for electronic ICs."

- Arduino.cc (05/02/2018)

5.1.6 Current Limits on the Arduino Nano 33 IoT

Maximum current per pin is 7mA
Maximum source current is 46mA
Maximum sink current per pin group is 65mA
Maximum current from 3.3V pin is 50mA

The MPM3610 DCDC converter that regulates voltage down to 3.3V has a maximum rating of 1.2A. However, "since there is no cooling area on the PCB this will be too much - especially at high VIN voltages (maximum rating is 21V)" - MPS (01/07/2015)

5.1.7 Prototype V1 Components and Power Budget

Part Description	Function	Power (mA)	QTY
Arduino Nano 33 IoT (3.3V, BLE, WiFi)	Microcontroller	<20	1
BLE module	Wireless communications	<=47	1
WiFi module	Wireless communications	<=112	1
Barcode/QR code reader module	Scanning of Barcodes and QR codes	<190	1
Liquid Crystal Display (LCD) module	Display of user instructions, status	<85	1
RGB LED (Common Cathode)	Device status notification	<18	1
Light Dependent Resistor (LDR)	Detect smartphone screen proximity		1
AM312 Passive Infrared Detector (PIR)	Human proximity detection	<20	1
Protoboard Breadboard	Prototype circuit connectivity	0	1
Thru-Hole (TH) Series Resistors	Current limiting to protect IO pins		1
TH Pull Up Resistors	Set IO pin default logic level to HIGH		0
TH Pull Down Resistors	Set IO pin default logic level to LOW		1
SMD Ceramic Capacitors	Decoupling / Bypass Capacitors	0	0
SMD Ferrite Beads Power Pin Filters	Filtering of high frequency noise.	0	0
	Total Power Budget	**<492**	**mA**

Prototype V1 - Parts and Their Functions (Table 4)

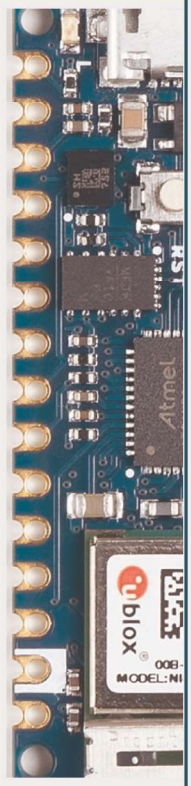
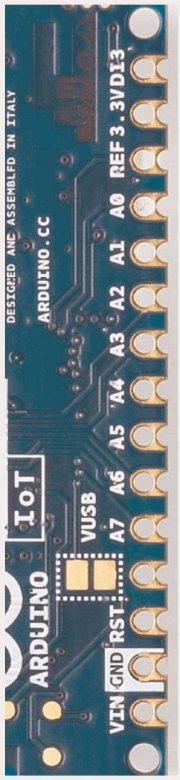
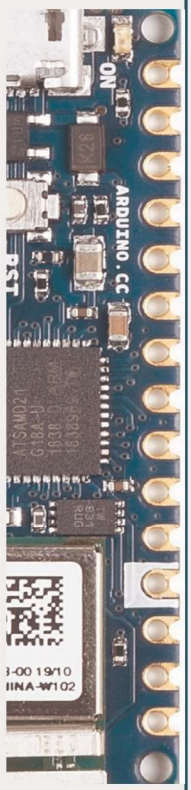
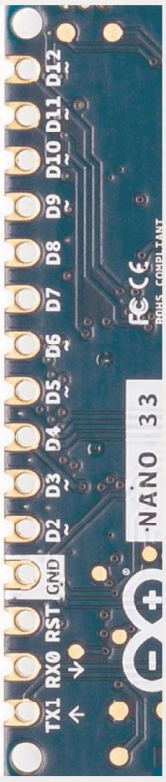
5.1.8 Adding More Serial Devices to the SAMD21 Microcontroller (SERCOM)

The SAMD21 Cortex-M0+ ARM MCU features SerialCommunication (SERCOM) technology which enables pins to be configured to map to specific peripheral buses such as UART and I2C. To use three serial ports on many other microcontrollers would require some of these ports to be implemented using Software Serial or "bit banging" techniques which can introduce reliability issues. However, with SERCOM pin mapping can be used to implement multiple hardware serial ports for connecting modules that feature serial interfaces.

	MKR Board Pin	Periph.C	Periph.D	Periph.G
		SERCOMx (x/PAD)	SERCOMx (x/PAD)	COM
00	D0	3/00	5/00	
01	D1	3/01	5/01	USB/SOF
02	D2	0/02	2/02	I2S/SCK0
03	D3	0/03	2/03	I2S/FS0
04	D4		4/02	I2S/MCK1
05	D5		4/03	I2S/SCK1
06	D6	5/02	3/02	I2S/SCK0
07	D7	5/03	3/03	I2S/FS0
	SPI			
08	MOSI	*1/00	3/00	
09	SCK	*1/01	3/01	
10	MISO	*1/03	3/03	I2S/SD0
	Wire			
11	SDA	*0/00	2/00	I2S/SD1
12	SCL	*0/01	2/01	I2S/MCK0
	Serial1			
13	RX		*5/03	
14	TX		*5/02	
16	A1		5/00	
17	A2		5/01	
18	A3		0/00	
19	A4		0/01	
20	A5		0/02	
21	A6		0/03	I2S/SD0
	ATWINC1501B SPI			
=26	WINC MOSI	*2/00	4/00	
27	WINC SCK	*2/01	4/01	
28	WINC SSN	2/02	4/02	
29	WINC MISO	*2/03	4/03	
	ATWINC1501B PINS			
32	WINC WAKE		4/00	
33	WINC IRQN		4/01	

Adding more Serial Devices to SAMD Microcontrollers Table 5

5.1.9 Prototype V1 - Arduino Nano 33 IoT Pinout Allocations

Arduino PCB		Arduino Nano 33 IoT		Connected Module / Component		
PCB Top	PCB Bottom	Pin Name	Config	Component	Pin Name	Config
						
		D13 (SI)	Serial2, RxFromLcd	NEXTIONLCD	TX	TX Output
		+3V3		QRSCAN, PIR		
		AREF				
		A0 D14				
		A1 D15				
		A2 D16~				
		A3 D17~				
		A4 D18				
		A5 D19~				
		A6 D20				
		A7 D21				
		+5V		NEXTIONLCD	5V	LCD VCC
		RESET				
		GND		NEXTIONLCD, QRSCAN, PIR, RGBLED		
		VIN				
PCB Top	PCB Bottom	Pin Name	Config	Component	Pin Name	Config
						
		~D12	PWM Output	RGBLED	R	Anode
		~D11				
		~D10	PWM Output	RGBLED	B	Anode
		~D9	PWM Output	RGBLED	G	Anode
		D8 (SI)	Serial2, TxToLcd	NEXTIONLCD	RX	RX Input
		D7				
		~D6 (S2)	Serial3, RxToSpareSerial	SPARE	TX	TX Output
		~D5 (S2)	Serial3, TxToSpareSerial	SPARE	RX	RX Input
		D4	Input, SenseFromPIR	PIR	PIR	Output
		~D3	Output, TriggerToQRScan	QRSCAN	TRIG	Input
		~D2	Input, StatusFromQRScan	QRSCAN	STAT	Output
		GND		NEXTIONLCD, QRSCAN, PIR, RGBLED		
		RESET				
		RX0 (S0)	Serial1, RX Input	QRSCAN		TX Output
		TX1 (S0)	Serial1, TX Output	QRSCAN		RX Input
		KEY: S0 = Serial (USB/HID), S1 = Serial1 (QRSCAN), S2 = Serial2 (NEXTIONLCD)				

Prototype V1 - Arduino Nano 33 IoT Pinout Allocations (Table 6)

5.1.10 USB Interface on the Arduino Nano 33 IoT

The USB connector of the board is directly connected to the USB host pins of the SAMD21. This routing enables use of the Arduino Nano 33 IoT as a client USB peripheral (acting as a mouse or a keyboard connected to the computer) or as a USB host device so that devices like a mouse, keyboard, or an Android phone can be connected to the Arduino Nano 33 IoT. This port can also be used as a virtual serial port using the **Serial** object in the Arduino programming language. The RX0 and TX1 pins are a second serial port available as **Serial1**.

5.1.11 Half-Duplex Communication via USB

Prototype V1 features an Arduino microcontroller emulating a keyboard when connected to a Tablet device configured as a USB host. Communication flows in one direction only from the Arduino to the Tablet. Bidirectional communication will be enabled in Prototype V2 using the WebUSB API for the Chrome web browser.

5.1.12 Serial Ports and USB on the Arduino Nano 33 IoT

SoftwareSerial.h is not required for the Arduino Nano 33 IoT because this board instead offers hardware serials that can be assigned to different pins.
This feature is enabled by the Atmel SAMD21G using what is called I/O multiplexing. The microcontroller provides 6 of what are known as SERCOM's (short for "SERial COMmunication") and these can be assigned to most pins.

SERCOM's that are by default already used by the Arduino Nano 33 IoT include:

- SERCOM2 for SPI NINA
- SERCOM3 for MOSI/MISO
- SERCOM4 for I2C bus
- SERCOM5 for Serial debugging (USB)

In addition, SERCOM0 and SERCOM1 are available to be assigned RX and TX pins to configure and enable an additional UART serial port.

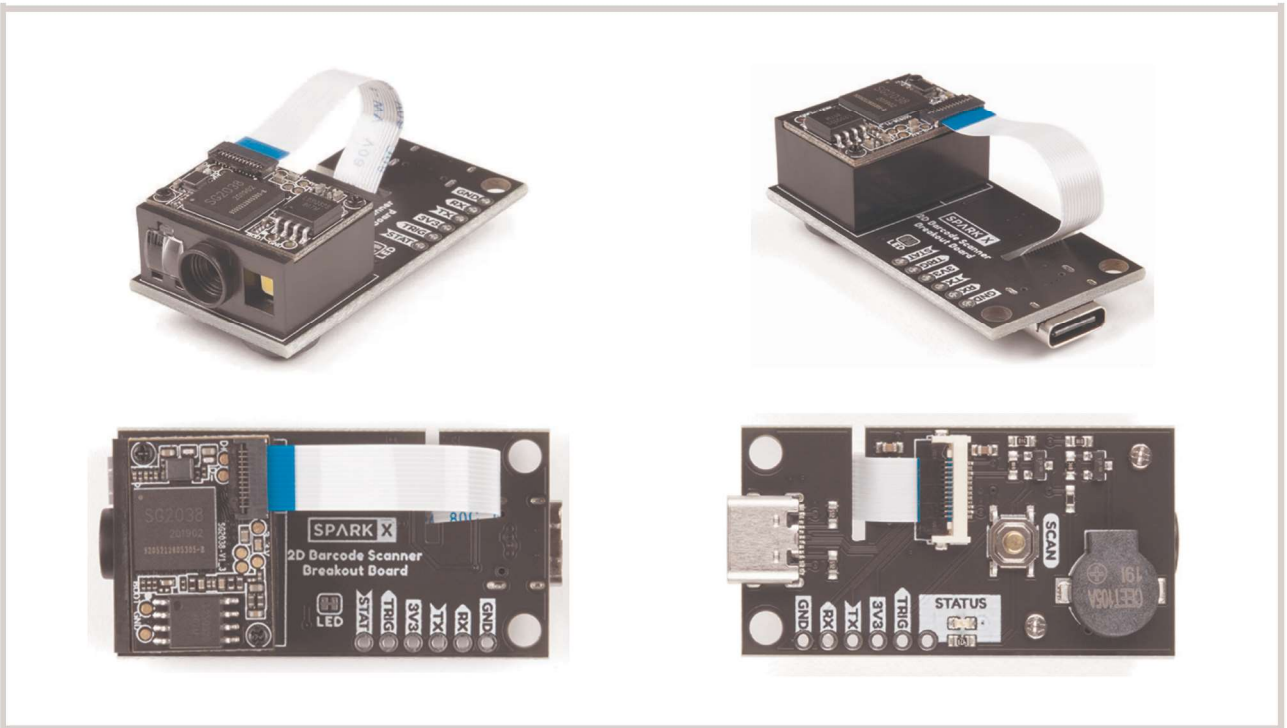
An Arduino Nano 33 IoT can be configured to use:

1. A natively hardware UART
2. A SERCOM assigned hardware UART
3. A USB port (host or client).

With the Arduino configured as a USB host a physical keyboard can interface directly with it sending input data only. With the Arduino configured as a USB client the Arduino can interface directly with a PC to emulate a keyboard by sending input data only.

5.1.13 Barcode and QR Code Reader Module

The SparkFun QR Code Reader Module features a DE2120 optical camera module for use in scanning ID barcodes as well as 2D/QR codes. The module is attached to a small 3.3V breakout PCB board with PTH holes for attaching a 2.57mm pitch thru-hole (TH) pin header. SparkFun have published a driver library for the module for use in controlling the camera and scanning features from a connected Arduino microcontroller.



SparkFun QR Code Reader Module with 3.3V Breakout Board (Figure 5)

Voltage and Current Limits

The QR Code Reader Module is powered from the 3.3V pin of the Arduino Nano 33 IoT and draws up to 190mA of current when continuous scanning is in use.

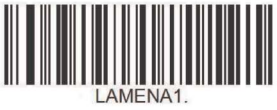
Pin #	Label	Arduino Pin	Function
1	GND	GND	0V
2	RX	~D6	RX serial line from UART connection on Arduino Nano 33 IoT.
3	TX	~D5	TX serial line to UART connection on Arduino Nano 33 IoT.
4	3V3	+3V3	VCC
5	TRIG	~D3	Continuous scanning mode is activated when logic level is HIGH.
6	STAT	~D2	Module Status.

QR Code Reader (J1) Module Header Pinout (Table 7)

5.1.14 Configuration of the Scanning Module

Setup and Configuration

The QR Code Reader Module can be set up by scanning specific configuration barcodes from the technical documentation and user manual. The following barcode settings were found to be particularly helpful for enabling reliable scanning.

 DEFALT. Restore default settings	 LAMENA1. *Turn on Illumination	 LAMENA0. Turn off Illumination
 PORVIC. USB-COM	 AIMENA1. *Trun on Aiming light	 AIMENA0. Trun off Aiming light
 POR232. Serial Default :Rate 9600bps; Data 8, Stop 1, Paity none.	 MIRLRE1. Trun on image Flipping (Support normal and flip)	 QRCENA1. *Enable QR Code
 SCMMAN. *Trigger Mode	 ODCENA. Enable 1D Symbologies	 AQRENA. Enable 2D Symbologies
 SCMCNT. Continuous scanning mode	 CNTALW3. Continuous output of the same code, 1 seconds interval	 SCMMDH. Sense Mode

Useful Barcode Reader Configuration Settings (Table 8)
Source: DY-Scan 2D Barcode Scanner Settings Manual

5.1.15 QR Code Scanning Process

USB Test Mode

The module features a USB-C port that enables a convenient way to test the ID and 2D code scanning capability of the module. This testing can be done by connecting a USB-host enabled PC to the module which then emulates a keyboard each time a code is scanned. The text characters of scanned codes are transmitted over the USB cable to the connected PC where the text will appear in an open text document just as it would if the characters were typed in via a connected keyboard.

Motion Sensor Triggering

The Arduino firmware sketch will be programmed so that when a connected AM312 PIR motion detector sensor is triggered by human presence the Arduino will then send a logic HIGH signal to the connected TRIG (Trigger) pin of the QR Code Reader Module. This signal then activates the continuous scanning mode of the module whereby the white LED will turn on illuminating the space the camera is pointed toward. A red light is also activated for the purpose of showing the user where to position the code they wish to scan.

Motion Sensor Deactivation

When the AM312 PIR sensor no longer detects human presence near the device then the Arduino firmware will send a logic LOW signal to the connected TRIG pin of the QR Code Reader Module. This signal will instruct the module to end the continuous scanning mode and turn off the white and red lights.

Successful Scanning

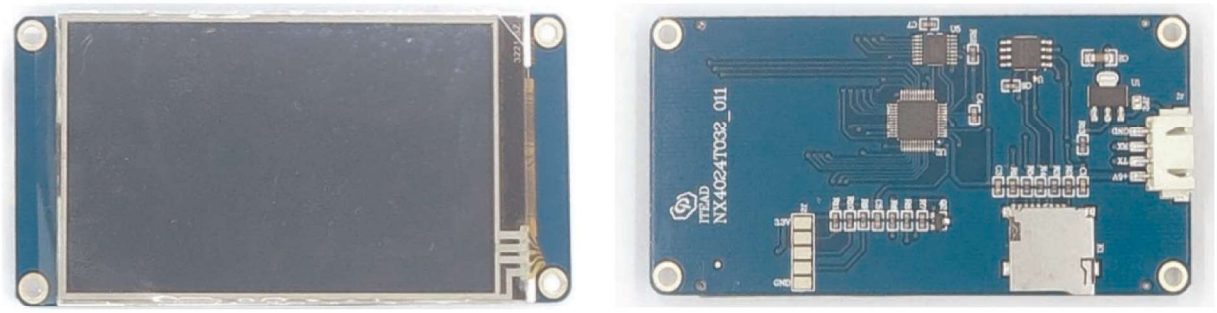
Alternatively, if a QR code is correctly presented and thus successfully scanned then a piezoelectric speaker on the module will beep to alert the user that the scan was made. The continuous scanning mode will then deactivate and the white and red lights will turn off. The text characters of the scanner code will be sent via the TX UART serial communication line to the connected input pin on the Arduino. At this point the Arduino will attempt to transmit the code to a connected Bluetooth device or to a server via an internet connected WiFi connection.

5.1.16 3.2" Nextion LCD Display

The Nextion NX4024T032_011R is a 3.2" LED LCD Display module featuring a resistive touchscreen reader. Although powered by a 5V source the TX and RX serial lines to and from the Arduino, the UART pins operate at 3.3V and are 5V tolerant. The LCD module can be expected to draw up to 85mA when not in a low power mode.

Pin #	Label	Arduino Pin	Function
1	VCC	+5V	5V
2	TX	D13	<3.3V TX serial line to UART on Arduino Nano 33 IoT. 5V tolerant.
3	RX	D8	<3.3V RX serial line from UART on Arduino Nano 33 IoT. 5V tolerant.
4	GND	GND	0V

3.2" Nextion LCD Display Pinout (Table 9)



3.2" Nextion LCD Display with Resistive Touch Screen Panel (Figure 6 a, b)

User Interface

A Software Development Kit (SDK) available from the manufacturer of the LCD module is used for designing a custom Graphical User Interface (GUI) that is interactive through use of the touch screen reader.

Firmware Commands and Instructions

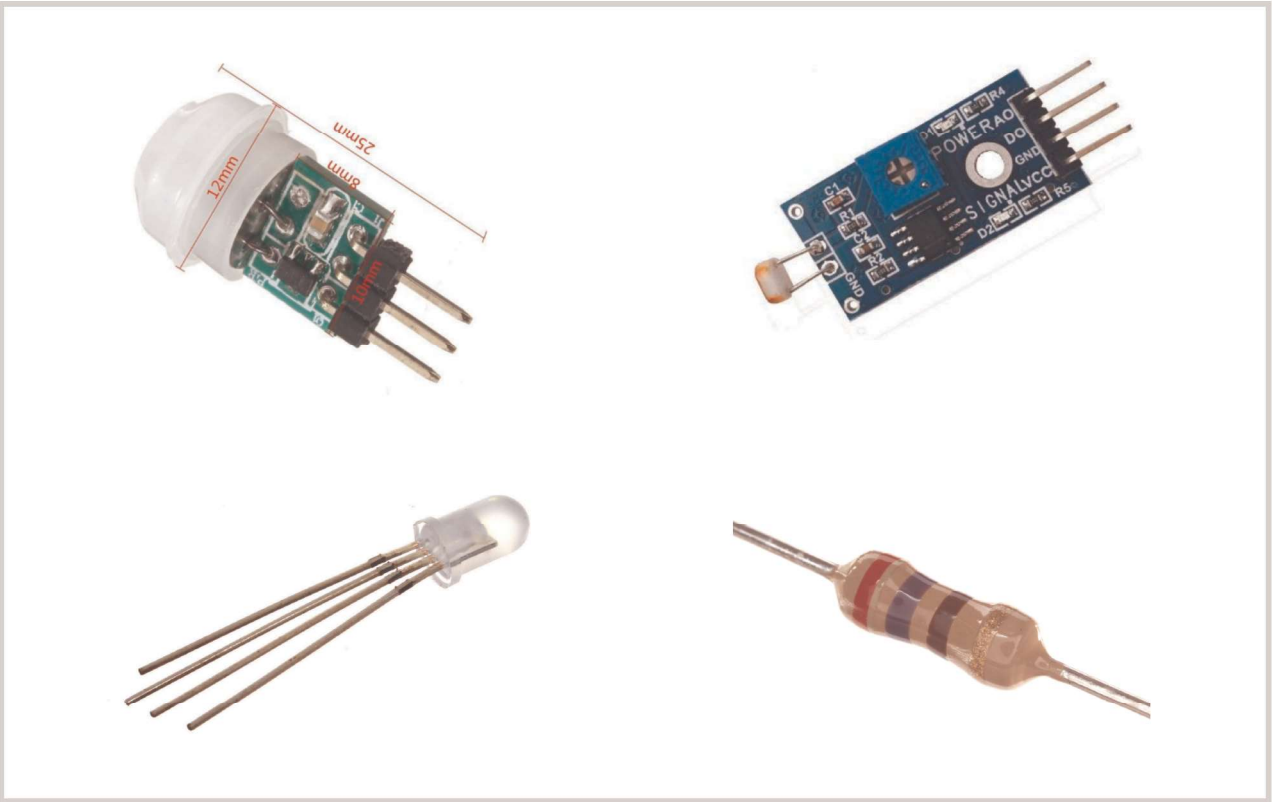
An Arduino Library for firmware development provides commands for exchanging data commands and messages between the Arduino microcontroller and the Nextion touch screen LCD controller.

Enclosure Parts

STL 3D model files are available from the manufacturer that allow 3D printing of Bezel and Base plates as parts of an enclosure for containing the LCD module. These parts can be used as inspiration for designing a larger enclosure capable of containing all components of the QR Code Scanning Kiosk.

5.1.17 Sensors and Actuators

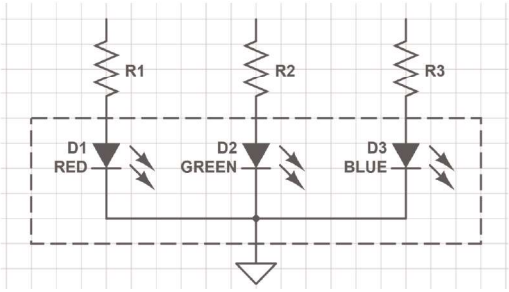
The components for Prototype V1 are connected using a Breadboard. Therefore, most of the components for Prototype V1 are of the Thru-Hole (TH) type. For other future Prototypes most of the components used will be of the Surface Mounted Device (SMD) type and will be either soldered directly to PCB circuit boards or, alternatively, interfaced with other components that are.



Other Components used in Prototype V1 (Figure 7 a, b, c, d) clockwise:
a: AM312 PIR Motion Sensor (3.3V power source, <20mA, typically <0.1mA)
b: Light Dependent Resistor (LDR) module
c: Common Cathode RGB LED
d: Thru-Hole Resistor suitable for breadboard prototype.

5.1.18 Current Limiting Series Resistors for Common Cathode RGB LED

Three Arduino pins are configured as Pulse Width Modulation (PWM) outputs, connected to Red, Green and Blue pins of the RGB LED. The longest LED pin is connected to GND as the RGB LED has a common cathode. The maximum permitted DC current per IO pin of the Nano 33 IoT is 7mA so suitable current limiting series resistors are required for each pin to prevent the current limit being exceeded.



Common Cathode RGB Light Emitting Diode (LED) (Figure 8 a, b)
a: Common Cathode RGB LED Pinout b: With Current Limiting Series Resistors

Pin#	Label	Arduino Pin	Function
1	LED_{Green}	~D9	LED_{Green} Forward Voltage Drop = 3.0 to 3.4 V or ~3.2V
2	GND	GND	0V
3	LED_{Blue}	~D10	LED_{Blue} Forward Voltage Drop = 3.0 to 3.4 V or ~3.2 V when V_{Supply} = 3.3V
4	LED_{Red}	~D12	LED_{Red} Forward Voltage Drop = 1.8 to 2.2 V or ~2.0 V

Common Cathode RGB LED Pinout. (Table 10)

From Ohm's Law R=V/I and so suitable values of current limiting series resistors are:

$R_{Red} = (V_{Supply} - LED_{Red}) / (LED_{Current})$
$= (3.3 - 2.0) / (6mA)$
$= 217\Omega$ (in E24 series closest is 240Ω)

$R_{Green} = (V_{Supply} - LED_{Green}) / (LED_{Current})$
$= (3.3 - 3.2) / (6mA)$
$= 1.7\Omega$ (no series resistor required)

$R_{Blue} = (V_{Supply} - LED_{Blue}) / (LED_{Current})$
$= (3.3 - 3.2) / (6mA)$
$= 1.7\Omega$ (no series resistor required)

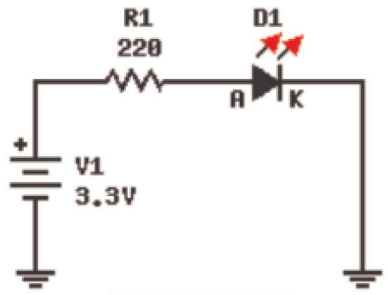
LED Color	Typical Vf Range
Red	1.8 to 2.1
Amber	2 to 2.2
Orange	1.9 to 2.2
Yellow	1.9 to 2.2
Green	2 to 3.1
Blue	3 to 3.7
White	3 to 3.4

Typical Forward Voltage Drops for various LED colours. (Figure 9)

Although 7mA is the maximum current rating for IO pins of the Arduino Nano 33 IoT, 6mA was used in the series resistor calculations so as to have a safety margin. The E24 values above are the next higher resistance values available within the E24 series of resistors.

A website tool can be used to check the series resistor calculated values:

Supply voltage: 3.3 V
LED current: 6 mA
Color, Voltage drop: red 2 V
Number of LEDs: 1
Resistor precision: 5% Calculate



Results:

Calculated resistance:	**216.67Ω**
Nearest lower 5% resistor:	**200Ω**
Nearest higher 5% resistor:	**220Ω***
Power dissipated by the resistor:	7.8 mW
Recommended resistor Wattage:	1/8 W
Power consumed by the LED (P_{LED}):	12 mW
Total power consumption (P_{Total}):	19.8 mW
Efficiency (P_{LED}/P_{Total}) x 100 :	60.61 %

10% 5% 2% 1%

*Nearest lower and higher **5%** resistors:*

Resistor color code	*Value*	*LED current*
	lower: ***200Ω***	*6.5 mA*
	higher: ***220Ω****	*5.91 mA*

LED Series Resistor Calculator used to check resistor calculations. (Figure 10)

5.1.19 Controlling RGB LED Combined Colours from Firmware with PWM

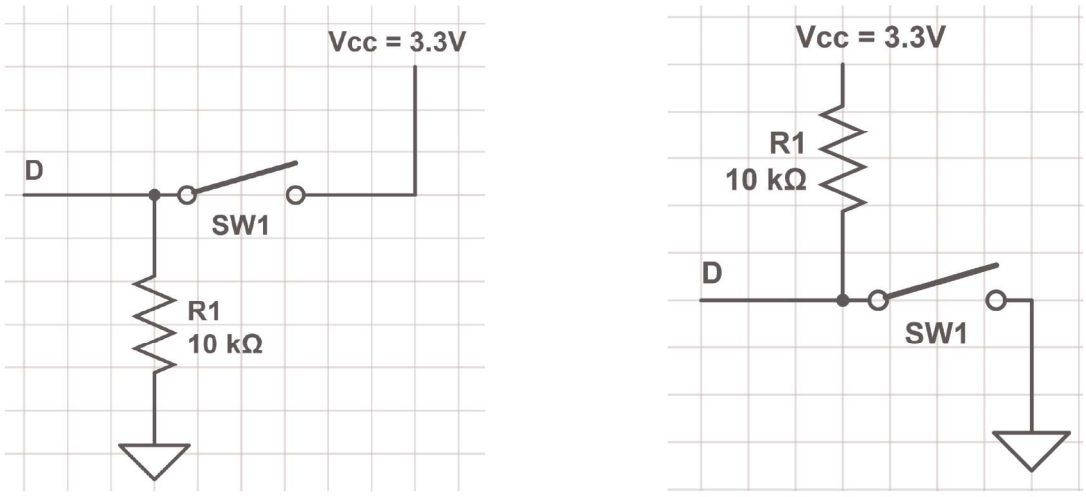
The Red, Green and Blue pins of the RGB LED are each connected in series with a current limiting resistor to a digital output pin that has Pulse Width Modulation (PWM) enabled. By mixing the PWM output values driving LED's for the three primary colours colour mixing will occur. PWM output values can be mixed to create a desired LED colour corresponding to a designated device status message that may be useful in troubleshooting operation of the device. Solid and flashing colours may have different status meanings. Proposed status definitions are shown in the table below:

Colour	Red PWM	Green PWM	Blue PWM	Status (if Solid LED)	Status (if Flashing LED)
Red	255	0	0	Waiting for the server.	Server connection error.
Orange	128	128	0	Reserved.	Reserved.
Green	0	255	0	Systems Nominal.	Loading, please wait.
Blue	0	0	255	Message sent.	New message received.

RGB LED Status Colours and their Corresponding PWM Output Values (Table 11)

5.1.20 Pull Up and Pull Down Resistors

The use of Pull Up and Pull Down Resistors enable input pins connected to sensors to be assigned default values to prevent input logic levels from floating or behaving unpredictably. The maximum permitted DC current per IO pin of the Arduino Nano 33 IoT is 7mA.



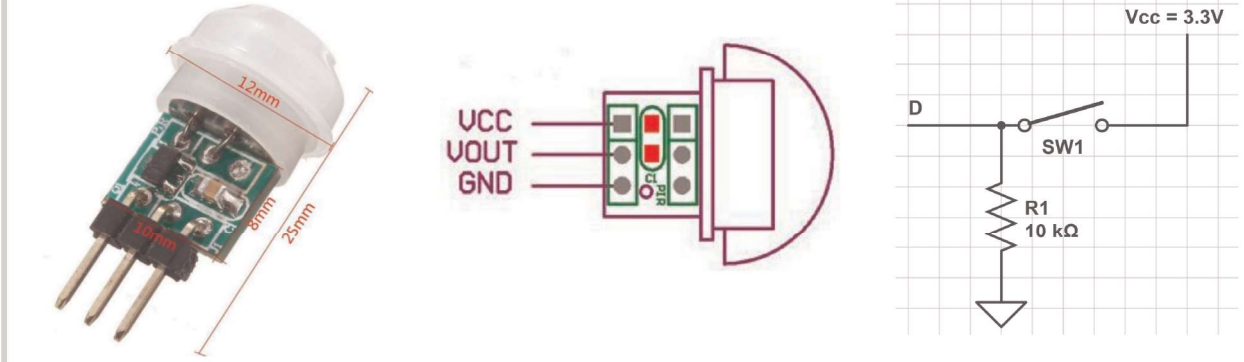
Pull Down and Pull Up Resistors on IO Pins (Figure II a, b)
a: Pull Down Resistor (left) on switched input pin (default logic LOW)
b: Pull Up Resistor (right) on switched input pin (default logic HIGH)

While Pullup and Pulldown resistors may be added to the circuit as components this is not necessary with the Arduino Nano 33 IoT board. This is because the board supports use of internal pullup and pulldown resistors that can be configured programmatically by the firmware.

5.1.21 Pyroelectric InfraRed (PIR) Motion Sensor

The AM312 PIR sensor module sources power from the 3.3V pin of the Arduino board and draws <20mA or <0.1mA when not triggered.
The V_{OUT} sense pin is fed into the Nano 33 IoT as a digital input. The input pin has an internal pulldown resistor configured in the Arduino firmware in order to set the V_{OUT} default untriggered logic state to LOW (i.e. no motion detected).
Once the triggered V_{OUT} logic state goes HIGH it will remain there until 2 seconds after human activity ceases to be detected. Retriggering can be enabled by setting the jumper to the H position.
The maximum permitted DC current per IO pin of the Arduino Nano 33 IoT is 7mA. Some PIR modules feature a potentiometer or jumpers which can be adjusted to preset the sensitivity and delay time when a motion detection event is triggered.

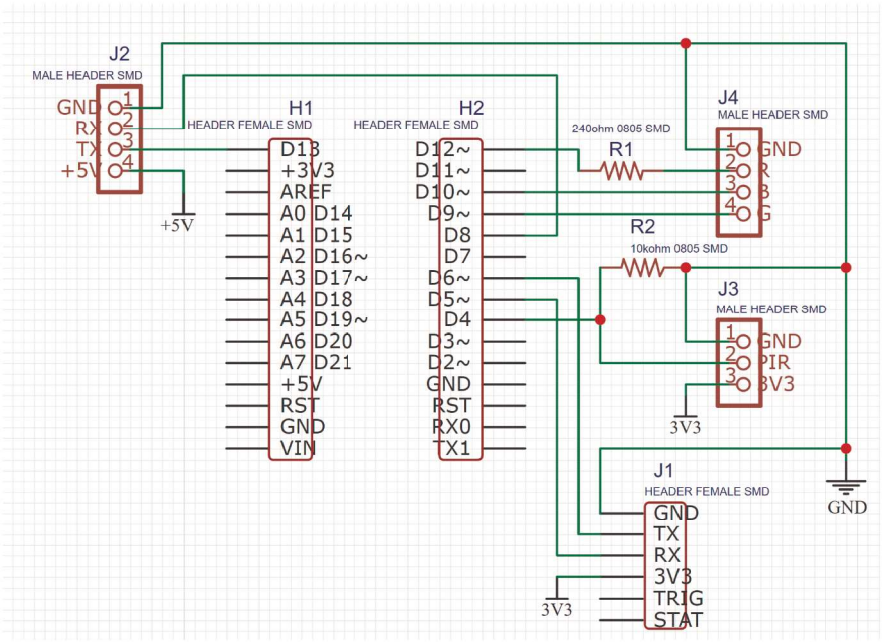
Pin #	Label	Arduino Pin	Function
1	V_{CC}	+3V3	3.3V
2	V_{OUT}	D4	Logic HIGH when triggered, remains HIGH until human exits the area.
3	GND	GND	0V
	Mini PIR Motion Sensor Pinout (Table 12)		



a: AM312 PIR motion sensor. b: Schematic. c: Pull Down resistor on Arduino input pin. Arduino Nano 33 IoT features configurable internal pulldown resistors. (Figure 12 a, b, c)

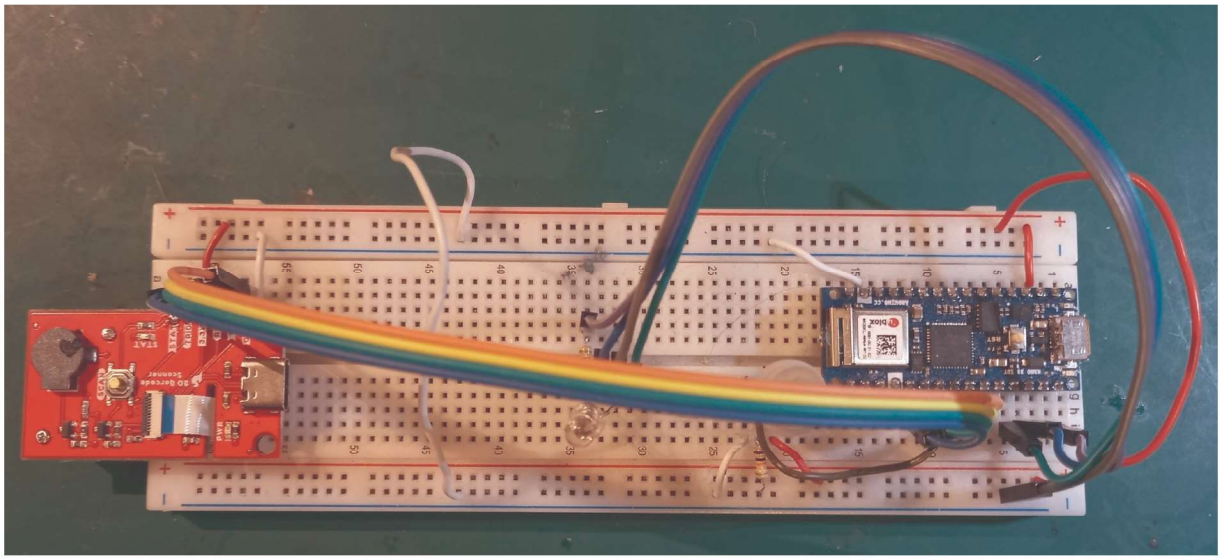
Feature	Function
Working Voltage	2.7 to 12 VDC
Static Power Consumption	<0.1mA
Delay Time	2 seconds.
Blocking Time	2 seconds.
Trigger	Can be repeated if the jumper has been set to the H position.
Sensing Range	<=100 degree cone angle, 3-5 m (depending on the lens)
Working Temperature	-20 to +60 Celsius
Mini PIR Motion Sensor Module (Table 13)	

5.1.22 Breadboard Prototype Circuit Schematic



Breadboard Prototype Circuit Schematic in EasyEDA (Figure 13)

5.1.23 Breadboard Circuit Arrangement



Breadboard Circuit Arrangement (Figure 14)

5.I.24 Third Party Firmware Libraries

Various third-party open source software libraries have been identified for inclusion in the Arduino firmware to provide additional features and functionality to the device without adding too much complexity to development of the firmware. The preferred Intellectual Property (IP) licence for third party code is the MIT Licence as this permits developers to reuse the code in their own commercial projects without payment of licensing fees or imposement of most other kinds of restrictions.

Code Library	Function and Comments	Prototype Version	IP Licence
<SparkFun_DE2120_Arduino_Library.h>	Barcode/QR code scanning. Modifications needed for SERCOM use.	1	GPL
"wiring_private.h"	Used for configuring SERCOM for enabling multiple hardware serial ports.	1	?
<EasyNextionLibrary.h>	Serial comms for Nextion LCD Module.	2	MIT
<trigger.h>	Installed with <EasyNextionLibrary.h>	2	MIT
<WiFiNINA.h>	WiFi comms for Nano 33 IoT board. Needed from Prototype V2 onwards for enabling Internet connectivity for calibrating internal Real-Time Clock (RTC) from internet-connected atomic clocks accessible via the Network Time Protocol (NTP).	3	?
<WebUSB_Arduino.h> TBC	Bidirectional comms between Arduino and website loaded in the Chrome web browser on a USB-connected Device.	>5	?
<RTCLib.h> by Adafruit	Library for the Adafruit DS3231 RTC module.	3	MIT
<NTPClient.h>, <EasyNTPClient.h>	Network Time Protocol (NTP) support for the RTC to be calibrated from an internet-connected atomic clock.	3	?
<ArduinoBLE.h>	Bluetooth/BLE communications. May not be needed when restricted to use of USB and/or WiFi connectivity.	>5	?

Third Party Firmware Code Libraries (Table I4)
A list of all third party firmware libraries with download links is in Appendix C.

5.I.25 Arduino Firmware Code for Prototype V1

sketch_jun14a §

```
void setup() {
  bool result = makeLEDRed();
  // put your setup code here, to run once:
  pinMode(SenseFromPIRPin, INPUT);
  pinMode(TriggerToQRScanPin, OUTPUT);
  pinMode(TxToQRScanPin, OUTPUT);
  pinMode(RxFromQRScanPin, INPUT);
  pinMode(RxFromLcdPin, INPUT);
  pinMode(TxToLcdPin, OUTPUT);
  pinMode(LEDredPin, OUTPUT);
  pinMode(LEDbluePin, OUTPUT);
  pinMode(LEDgreenPin, OUTPUT);
  Serial.begin(9600);
  while(!Serial) {
    ; //wait for USB serial port to connect
  }
  Serial1.begin(9600);
  Serial.println("wait to connect to scanner serial port");
  while(!Serial1) {
    ; //wait to connect to scanner serial port
  }
  Serial2.begin(9600);
  Serial.println("wait to connect to LCD serial port");
  while(!Serial1) {
    ; //wait to connect to scanner serial port
  }
  bool result = makeLEDGreen();
}

void loop() {
  // put your main code here, to run repeatedly:
  val = digitalRead(SenseFromPIRPin);
  if(val == HIGH) {
    modeScan = true;
    bool result = makeLEDBlue();
    Serial.println("HIGH");
  } else {
    modeScan = false;
    Serial.println("LOW");
  }
  if(modeScan) {
    scanner.startScan();
    Serial.println("startScan()");
    //digitalWrite(scannerTriggerPin,HIGH);
  } else {
    scanner.stopScan();
```

Arduino Firmware Code for ProtoType V1 (Figure 15)

Complete Firmware Code for Prototype V4 is included in **Appendix B.**

5.2 Prototype V2 - Add Adapter PCB V1, LCD PCB

5.2.1 Prototype V2 Goals

Prototype V1 has implemented on a wired breadboard circuit most of the modules and components intended for use in the project and was a functional proof of concept.

Objectives for Prototype V2 include:

1. Design, manufacture, assemble and test a custom PCB Breakout Board (PCB V1) interfaced with the microcontroller.

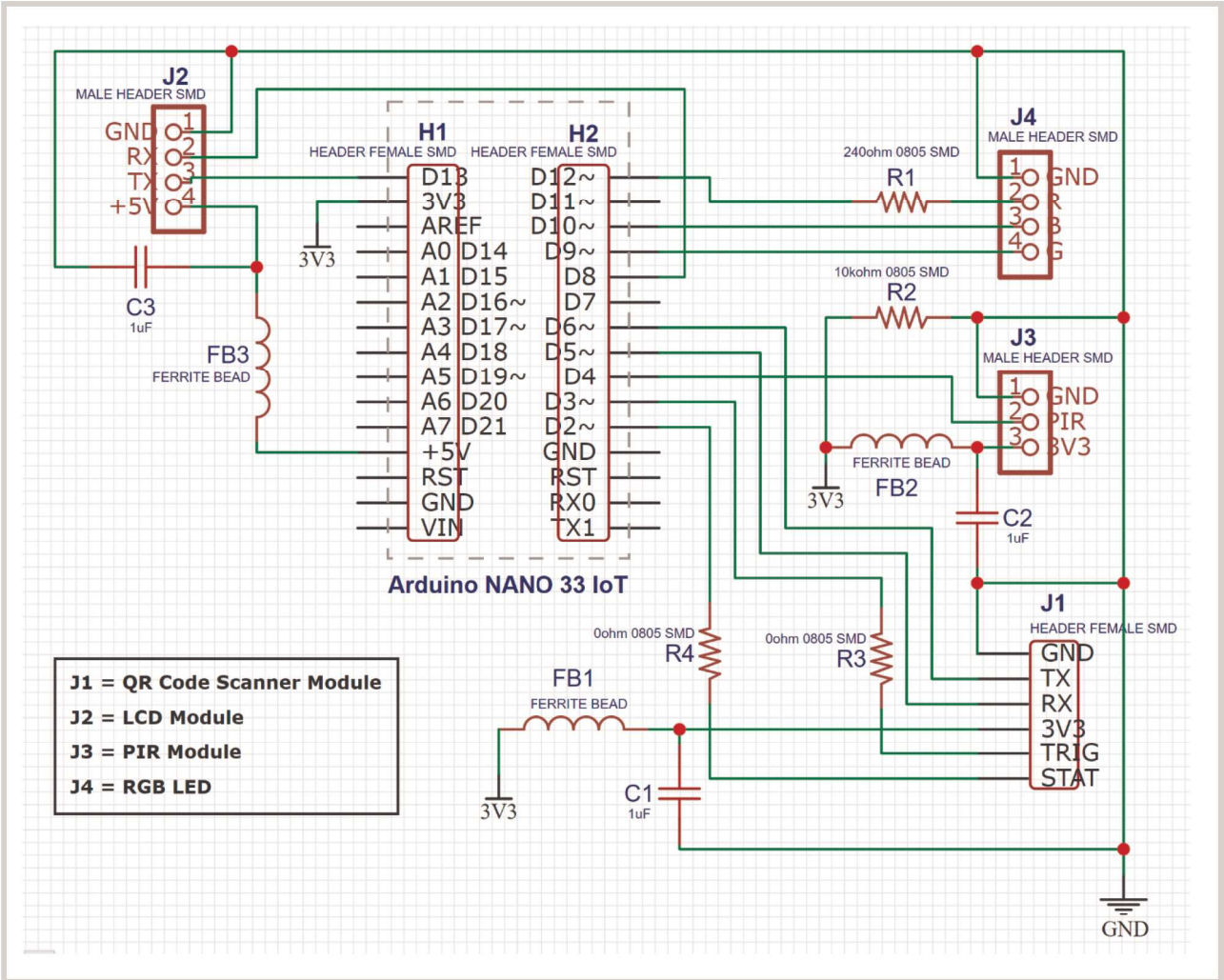
5.2.2 Prototype V2 Components and Power Budget

Part Description	Function	Power (mA)	Quantity
Arduino Nano 33 IoT (3.3V, BLE, WiFi)	Microcontroller.	<20	1
BLE module	Communications.	<=47	NC
WiFi module	Communications.	<=112	NC
Barcode/QR code reader module	Barcode and QR code scanning	<190	1
DS3231 Real-Time Clock (RTC)	**Time and Date tracking. (Mk2 only)**		**1**
3V CR1220 Lithium Coin Battery for RTC	**Time and Date tracking. (Mk2 only)**	**NA**	**1**
3.7V 1200mAh LiPo Battery	**Rechargeable Battery.**	**NA**	**1**
RGB LED indicator	Device status notification	<18	1
AM312 Passive Infrared Detector (PIR)	Human proximity detection	0	0
Component interface adapter PCB	**Interfacing of device components**	0	**1**
SMD Series Resistors	Current limiting to protect IO pins	0	0
SMD Pull Up Resistors		0	0
SMD Pull Down Resistors		<1	1
SMD Ceramic Capacitors	**Decoupling / Bypass Capacitors**		3
SMD Ferrite Bead Power Pin Filters	**Protection from high frequency noise**		3
3D Printed Front Panel	**Bezel for LCD Module**	0	1
Contingency		**100**	**mA**
	Total Power Budget	**573**	**mA**

Prototype V2 - Parts and Their Functions (Table 15)

GREEN coloured text refers to components added in Prototype V2.
Note: WiFi and BLE normally can't be used simultaneously unless the user has hacked the device. Hacking the device in this way will void the warranty.

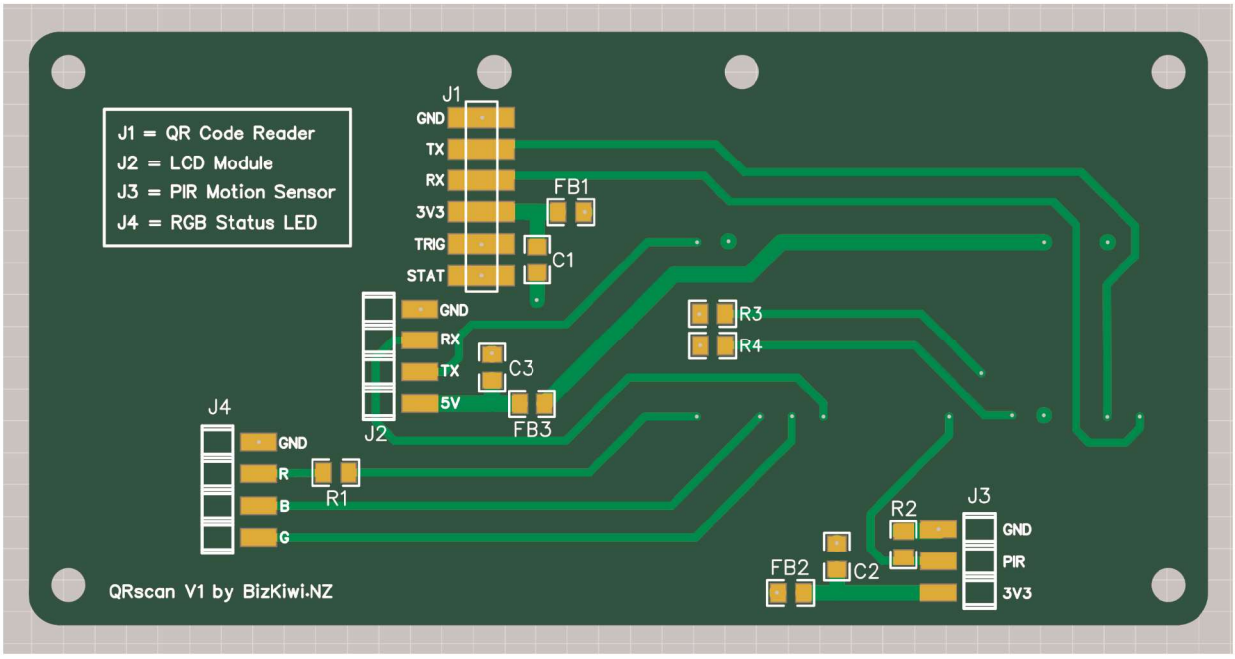
5.2.3 Prototype V2 - Custom PCB Circuit Schematic V1

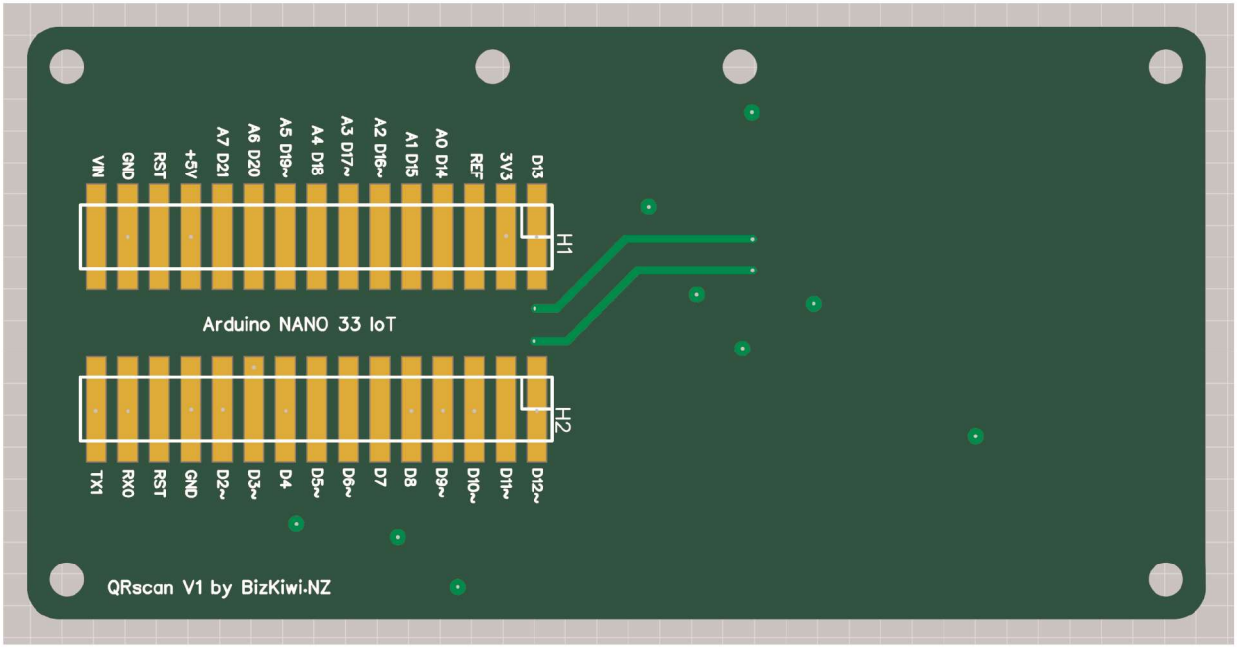


Prototype V2 - Custom PCB Circuit Schematic V1 in EasyEDA (Figure 16)

Decoupling Capacitors and Ferrite Bead Filters were added in this prototype.

5.2.4 Prototype V2 - Custom PCB V1 Rendered Design





Custom PCB V1 Design as Rendered in EasyEDA

Dimensions: 95mm x 47.5mm
The PCB dimensions are the same as the Nextion LCD PCB.
This is so the 2 boards can be stacked and separated by nylon spacers.

(Figure 17 a: Top Layer, b: Bottom Layer)

5.2.5 Prototype V2 - Custom PCB V1 Pin Headers

Designator	Module / Component	Interface	Pins	Footprint	PCB Side
H1	Arduino - Left side pin header	NA	15	SMD 2.54mm	Bottom
H2	Arduino - Right side pin header.	NA	15	SMD 2.54mm	Bottom
J1	QR Code Reader Module	UART	6	SMD 2.54mm	Top
J2	LCD Module	UART	4	SMD 2.54mm	Top
J3	PIR Module	Digital IN	3	SMD 2.54mm	Top
J4	RGB LED (Common Cathode)	3 x PWM	4	SMD 2.54mm	Top

PCB V1 Pin Headers (Table 16)

5.2.6 Prototype V2 - Custom PCB V1 Passive Components

Designator	Component	Value	Footprint	PCB Side
R1	Resistor 10k ohm	10k ohm	SMD 0805	Top
R2	Resistor 240 ohm	240 ohm	SMD 0805	Top
R3	Resistor 0 ohm	0 ohm	SMD 0805	Top
R4	Resistor 0 ohm	0 ohm	SMD 0805	Top
FB1	Ferrite Bead Power Pin Filter	47 ohm @ 100MHz	SMD 0805	Top
FB2	Ferrite Bead Power Pin Filter	47 ohm @ 100MHz	SMD 0805	Top
FB3	Ferrite Bead Power Pin Filter	47 ohm @ 100MHz	SMD 0805	Top
C1	Decoupling / Bypass Capacitor (Ceramic)	0.1uF +-10% 10V	SMD 0805	Bottom
C2	Decoupling / Bypass Capacitor (Ceramic)	0.1uF +-10% 10V	SMD 0805	Bottom
C3	Decoupling / Bypass Capacitor (Ceramic)	0.1uF +-10% 10V	SMD 0805	Bottom

PCB Passive Components (Table 17)

5.3 Prototype V3 - Upgrade MCU, PCB, Add Lipo Battery, RTC, NTP

5.3.1 Troubleshooting Prototype V3

Prototype V2 featured a low cost microcontroller, the Arduino Nano 33 IoT, which was found to have undocumented reliability issues for supporting multiple UART hardware serial ports. The serial ports were configured following documentation instructions and a tutorial walk-through with negative results.

A great deal of effort and resources were put into troubleshooting the technical issues with the Nano 33 IoT. A second Nano 33 IoT board was purchased and tested with negative results. A newer version of the Barcode scanning module was also purchased and tested, again with negative results.

When the SERCOM configuration tutorial was followed with a MKR WiFi 1010 board the configured UART hardware serial ports functioned properly immediately and without any issues. The Nano 33 IoT is considered to be very similar in specifications to the larger MKR WiFi 1010 board.

The main differences are in that the MKR WiFi 1010 also features:

- ~+50% or ~NZ$10 increase in cost of the board.
- Added I2C interface connector port.
- Lithium Polymer (LiPo) rechargeable battery port and charging circuit.
- Stackable pin headers for addition of third party PCB shields.

5.3.2 Prototype V3 Goals

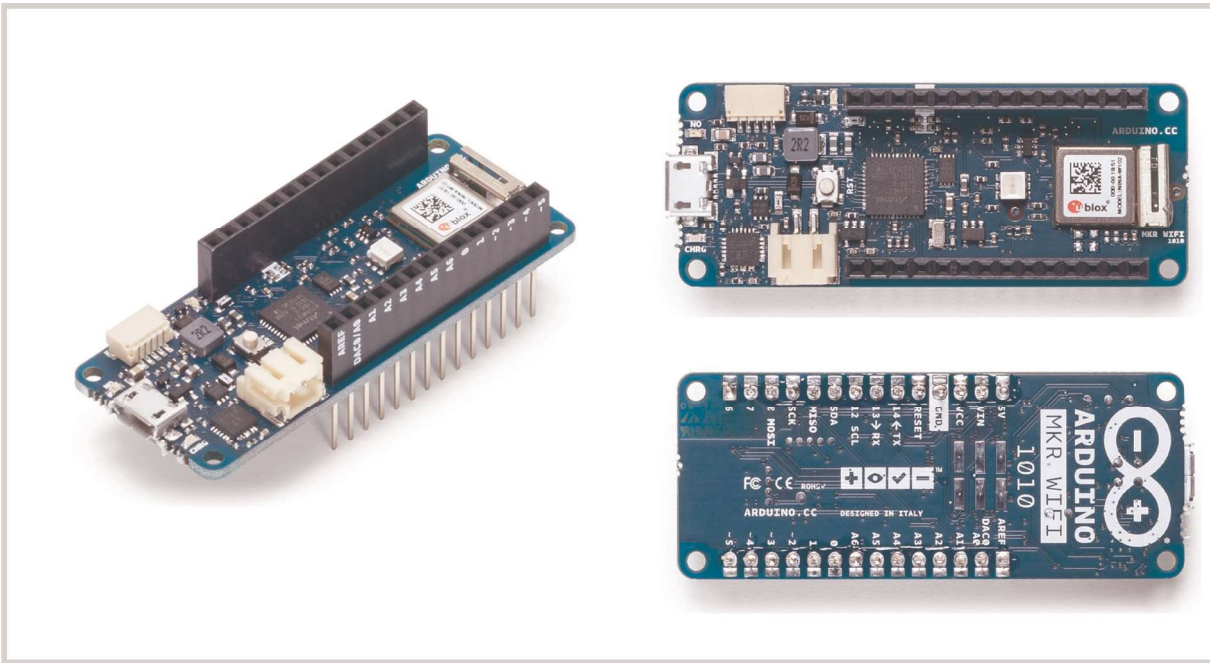
With increased pressure from time constraints, for Prototype V3 the decision was made to modify the prototype design by swapping the Nano 33 IoT board for instead the MKR WiFi 1010 board. Also added was a 3.7V 1200mAh LiPo rechargeable battery and an Adafruit DS3231 Real-Time Clock (RTC) module with backup battery..

Changes to the custom adapter PCB V2 were needed to accommodate the larger footprint of the MKR series board and its different pin header pinout interfacing.
It's important to note the fact that most of the MKR series of Arduino microcontrollers feature the same pinout specifications. This means the updated custom adapter PCB V2 can interface with any of the MKR compatible boards.

These MKR boards and battery arrangements can be used with the Adapter PCB V2:

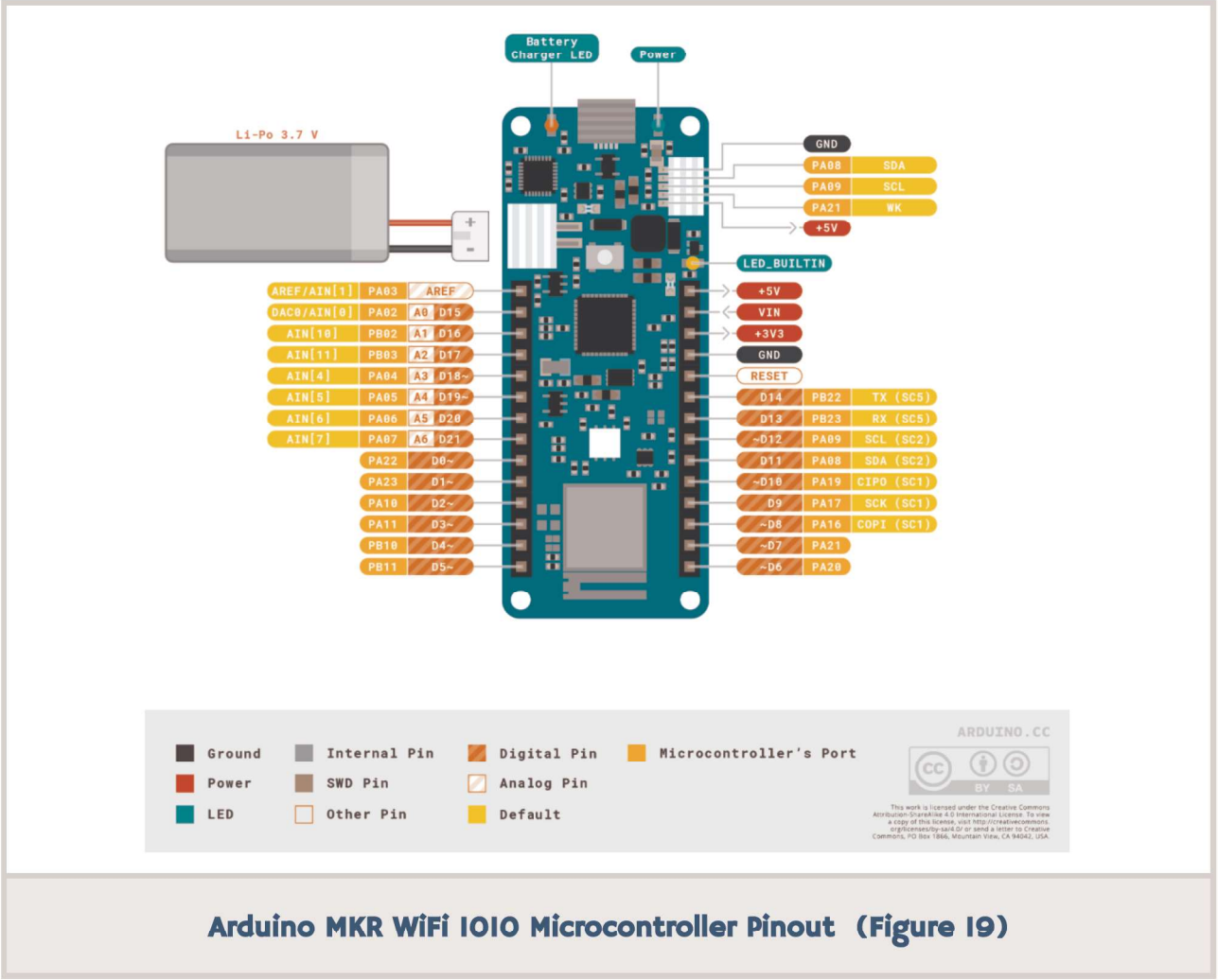
- MKR WiFi 1010 with 3.7V 1024mAh minimum Li-Po Single Cell Battery.
- MKR GSM 1400 with 3.7V 2500mAh minimum Li-Po Single Cell Battery.
- MKR NB 1500 with 3.7V 1500mAh minimum Li-Po Single Cell Battery.
- MKR WAN 1310 with 3.7V 1024mAh minimum Li-ion/Li-Po Single Cell Battery.
- MKR FOX 1200 with 2 x 1.5V AA or 2 x 1.5V AAA Batteries.

5.3.3 Arduino MKR WiFi 1010 Microcontroller



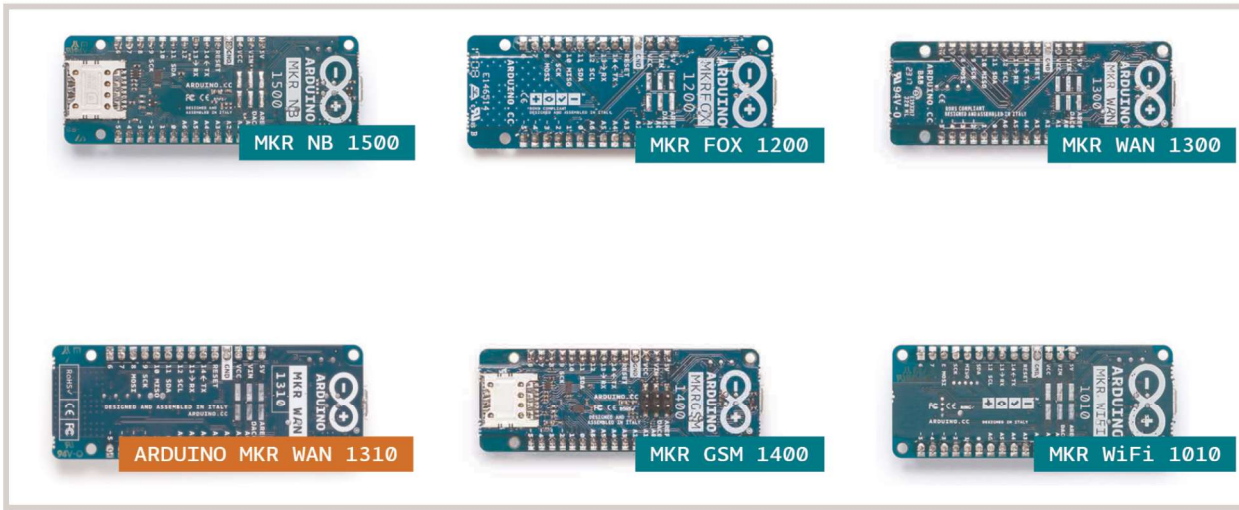
Arduino MKR WiFi 1010 Microcontroller Board (Figure 18 a, b, c)

5.3.4 Arduino MKR WiFi 1010 Microcontroller Pinout



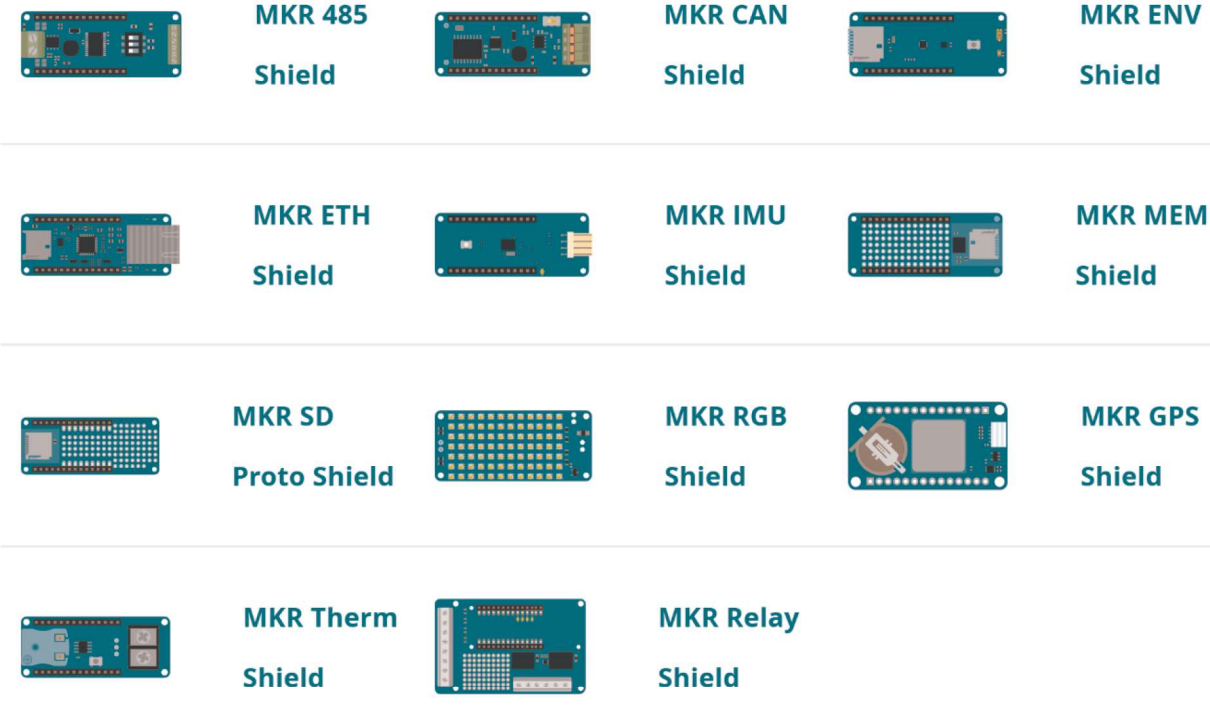
Arduino MKR WiFi 1010 Microcontroller Pinout (Figure 19)

5.3.5 Arduino MKR Boards Compatible with PCB V2 Custom Adapter PCB



MKR Microcontroller Boards pin compatible with PCB V2 Custom Adapter PCB. (Figure 20)

5.3.6 Arduino Stackable Shields Compatible with the MKR Series of Boards



Official Arduino stackable shields compatible with MKR Boards. Many third-party shields are also available. (Figure 21)

5.3.7 Arduino MKR WiFi 1010 Specifications

Feature / Function	Specification
Microcontroller	SAMD21 Cortex-M0+ 32bit low power ARM MCU
Radio Module	U-blox NINA-W102
Secure Element	ATECC508A
Supported Battery	LiPo Single Cell, 3.7V, 1024mAh Minimum
Clock	32.768 kHz (RTC), 48 MHz
Memory	CPU Flash Memory: 256KB, SRAM: 32KB, EEPROM: None
Connectivity	WiFi, BLE 4.2, USB (Native in the SAMD21 Processor)
Digital I/O Pins	8
PWM Pins	13 (0..8, 10, 12, 18 / A3, 19 / A4)
Interfaces	USB (Full speed USB Device and embedded Host), SPI, I2C, I2S, UART
Analog Input Pins	7 (ADC 8/10/12 bit)
Analog Output Pins	1 (DAC 10 bit)
External Interrupts	10 (0, 1, 4, 5, 6, 7, 8, 9, 16 / A1, 17 / A2)
Voltages	Input USB: 5V, Input VIN: 5V (6V max), Operating: 3.3V
Currents	DC per IO Pin Limit: 7mA
Pinout	Digital: 8, PWM: 13, Analog In: 7 (ADC 8/10/12 bit), Analog Out: 1 (DAC 10 bit)
Dimensions	25 x 61.5 mm (32 gm with headers)

Arduino MKR WiFi 1010 - Technical Specifications (Table 18)

5.3.8 Prototype V3 - Custom PCB V2 Pin Headers

Designator	Module / Component	Interface	Pins	Footprint	PCB Side
H1	Arduino - Left side pin header	NA	15	SMD 2.54mm	Bottom
H2	Arduino - Right side pin header.	NA	15	SMD 2.54mm	Bottom
J1	QR Code Reader Module	UART	6	SMD 2.54mm	Top
J2	LCD Module	UART	4	SMD 2.54mm	Top
J3	PIR Module	Digital IN	3	SMD 2.54mm	Top
J4	RGB LED (Common Cathode)	3 x PWM	4	SMD 2.54mm	Top
J5	DS3231 Real-time Clock (RTC)	I2C	8	SMD 2.54mm	Top

PCB V2 Pin Headers (Table 19)
GREEN = new addition to Prototype V3

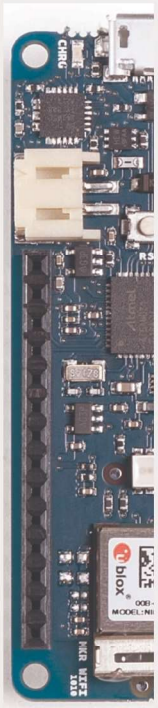
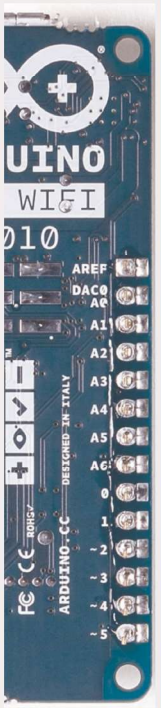
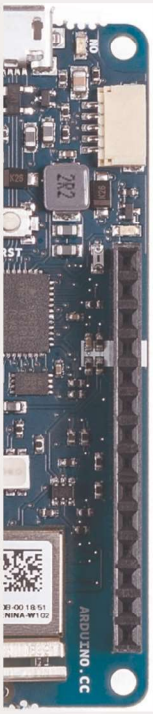
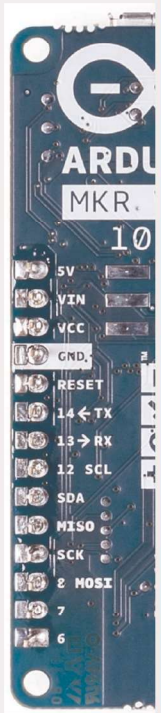
5.3.9 Components and Power Budget

Part Description	Function	Power (mA)	Quantity
Arduino MKR WiFi 1010 (3.3V, BLE, WiFi)	Microcontroller.	<20	1
BLE module.	Communications (can be disabled).	<=47	NC
WiFi module.	Communications (can be disabled).	<=112	NC
DS3231 Real-Time Clock (RTC) module	**Time and Date tracking.**		1
3V CR1220 Lithium Coin Battery for RTC	**Time and Date tracking.**	**NA**	1
3.7V 1200mAh LiPo Battery	**Rechargeable Battery.**	**NA**	1
Barcode/QR code reader module	Barcode and QR code scanning	<190	1
RGB LED indicator	Device status notification	<18	1
AM312 Passive Infrared Detector (PIR)	Human proximity detection	<20	1
Component interface adapter PCB V2	**Interfacing of device components**	**0**	1
SMD Series Resistors	Current limiting to protect IO pins	<5	3
SMD Pull Up Resistors	Ties logic high instead of floating.	0	0
SMD Pull Down Resistors	Ties logic low instead of floating.	<1	1
SMD Ceramic Capacitors	**Decoupling / Bypass Capacitors**		4
SMD Ferrite Bead Power Pin Filters	**Protect from high frequency noise**		4
3D Printed Front Panel	**Bezel for LCD Module**		
Contingency		**100**	**mA**
	Total Power Budget	**~598**	**mA**

Prototype V3 - Parts and Their Functions (Table 20)

GREEN coloured text refers to components added in Prototype V3.
Note: WiFi and BLE normally can't be used simultaneously unless the user has hacked the device. Hacking the device in this way will void the warranty.

5.3.10 Prototype V3 (PCB V2) - Arduino MKR WiFi 1010 Pinout Allocations

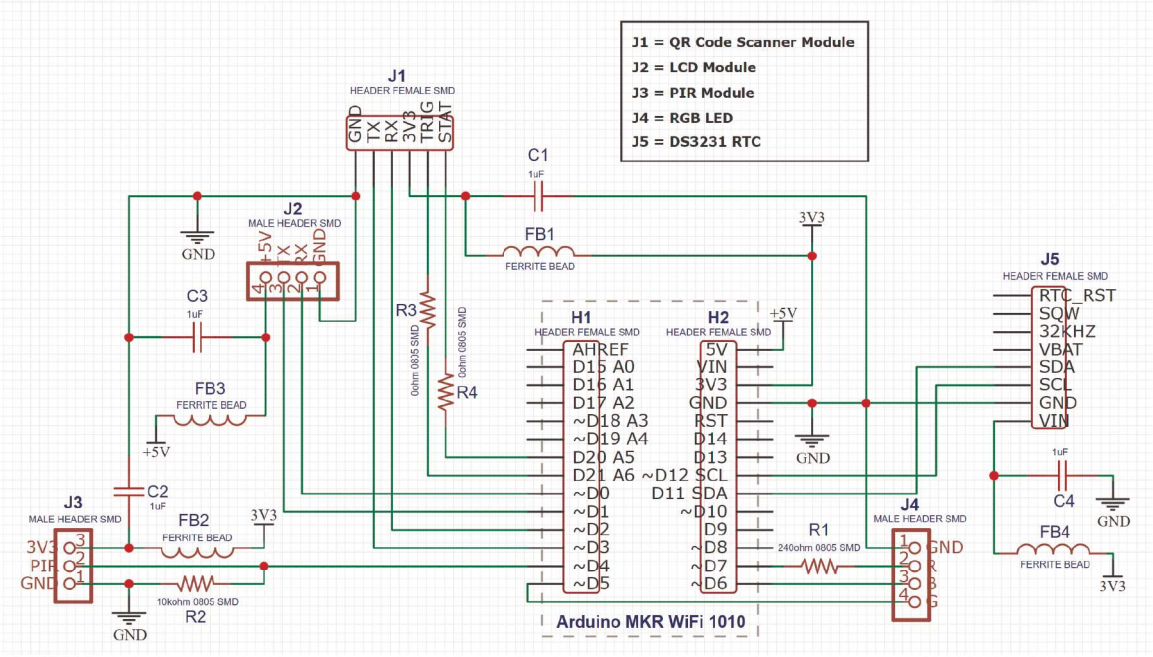
Arduino PCB			Arduino MKR WiFi 1010	Connected Module / Component		
PCB Top	PCB Bottom	Pin Name	Config	Component	Pin Name	Config
						
		AREF	Reserved for Shields			
		DI5 A0 DAC0	Reserved for Shields			
		DI6 AI	Reserved for Shields			
		DI7 A2	Reserved for Shields			
		~DI8 A3	Reserved for Shields			
		~DI9 A4	Reserved for Shields			
		D20 A5	Output, TriggerToQRScan	QRSCAN	TRIG	Input
		D2I A6	Input, StatusFromQRScan	QRSCAN	STAT	Output
		~DO	SERCOM3	NEXTIONLCD	RX	RX Input
		~DI	SERCOM3	NEXTIONLCD	TX	TX Output
		~D2	SERCOMO	QRSCAN	RX	RX Input
		~D3	SERCOMO	QRSCAN	TX	TX Output
		~D4	Input, SenseFromPIR	PIR	PIR	Output
		~D5	PWM Output	RGBLED	GREEN	Anode
PCB Top	PCB Bottom	Pin Name	Config	Component	Pin Name	Config
						
		5V		NEXTIONLCD		
		VIN				
		VCC (3V3)		QRSCAN, PIR, RTC		
		GND		NEXTIONLCD, QRSCAN, PIR, RGBLED, RTC		
		RESET				
		DI4 TX	SERCOM5 (Reserved)	SPARE UART	RX	RX Input
		DI3 RX	SERCOM5 (Reserved)	SPARE UART	TX	TX Output
		~DI2 SCL	I2C Bus	DS3231 RTC		
		DII SDA	I2C Bus	DS3231 RTC		
		~DIO MISO	Reserved for Shields			
		D9 SCK	Reserved for Shields			
		~D8 MOSI	Reserved for Shields			
		~D7	PWM Output	RGBLED	RED	Anode
		~D6	PWM Output	RGBLED	BLUE	Anode

KEY: S0 = Serial (USB/HID), S1 = Serial1 (QRSCAN), S2 = Serial2 (NEXTIONLCD)

Prototype V3 (PCB V2) - Arduino MKR WiFi 1010 Pinout Allocations (Table 21)

```
//Uart mySerial0 (&sercom0, 3, 2, SERCOM_RX_PAD_3, UART_TX_PAD_2); // RX=3, TX=2
//Uart mySerial3 (&sercom3, 1, 0, SERCOM_RX_PAD_1, UART_TX_PAD_0); // RX=1, TX=0
```

5.3.II Prototype V3 - Custom PCB V2 Circuit Schematic



Prototype V3 - Custom PCB Circuit Schematic V2 in EasyEDA (Figure 22)

Decoupling Capacitors, Ferrite Beads, RTC have been added since Prototype V1.

The types and values of passive electronics components were selected in part by previous learning including the following range of criteria:

"Decoupling Capacitors connect to ground in parallel with IC power supply pins and are used to maintain supply voltage to IC pins in events of other loads being turned on.

Ceramics were chosen for decoupling capacitors as Electrolytics have a particularly large footprint and Tantalums feature an Equivalent Series Resistance (ESR) that may be too low for the application.

Ferrite Beads as filters are placed in series with power pins on IC's to protect them from high frequency noise.

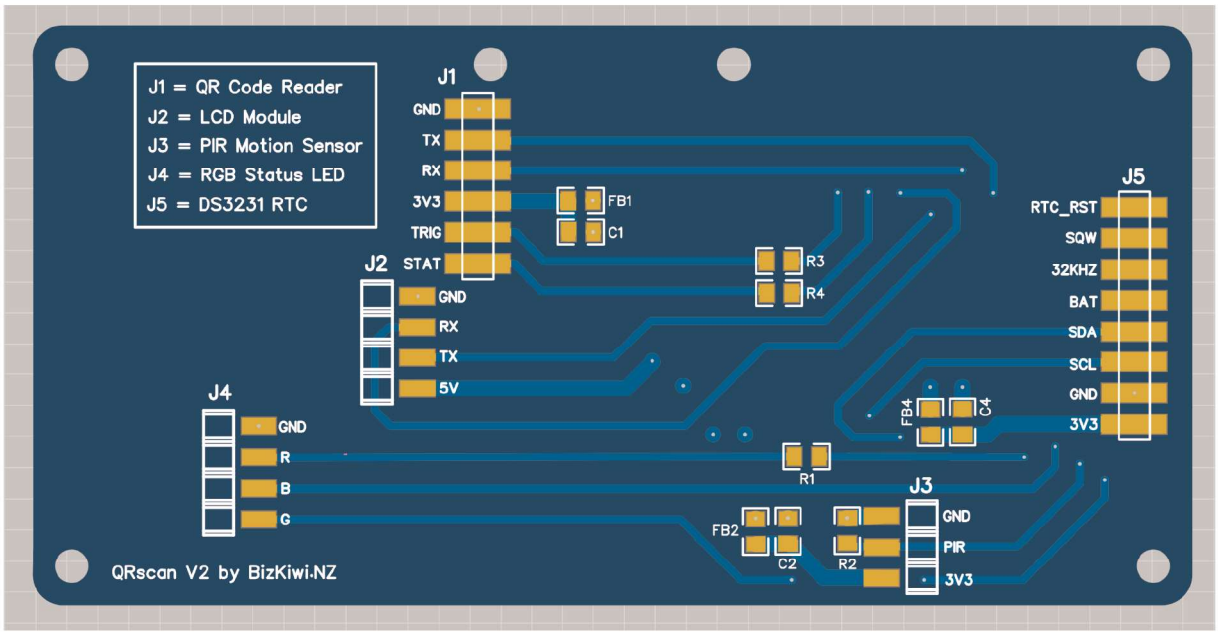
Resistors were selected from the 1/8W E24 series with surface mount 0805 (Imperial) packages.

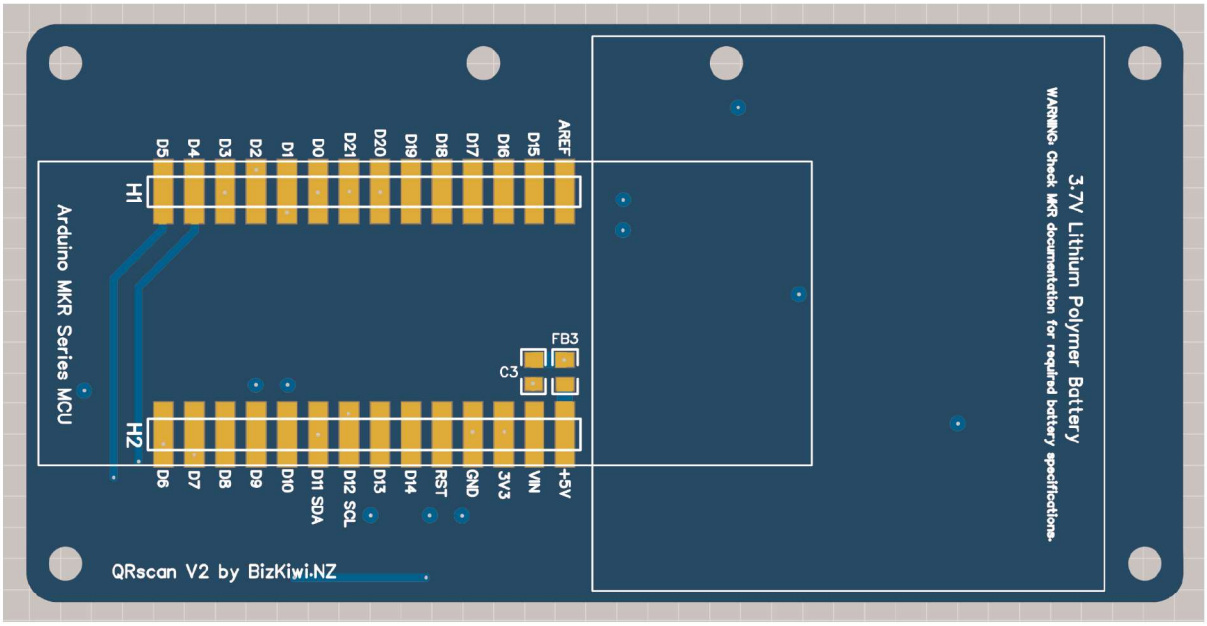
The 0805 SMD package size was selected for these reasons:

1. *The scale is sufficiently large enough to comfortably solder manually.*
2. *The scale is sufficiently large enough to allow a track to pass underneath SMD components (however, passing an analogue track under a digital track is not recommended).*
3. *The scale is sufficiently small enough (footprints and solder pads) to allow substitution of 0805 packages with the smaller 0602 packages without requirement for reworking or modifications to the PCB."*

Source: Julian, Stephen (May 2018). *DE6408 Electronics Manufacturing 2 Project.* Retrieved from author's personal Unitec academic archive.

5.3.12 Prototype V3 - Custom PCB V2 Rendered Design





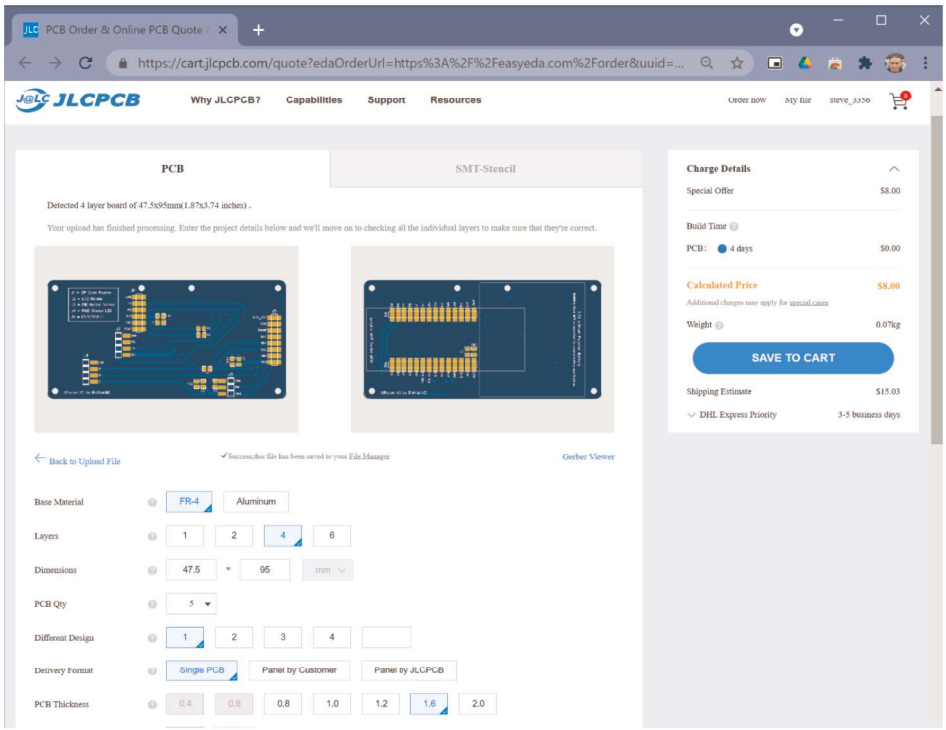
Prototype V3 - Design of Custom PCB V2 as Rendered in EasyEDA
A header for the Adafruit DS3231 RTC module has been added to this PCB version.

PCB Dimensions: 95mm x 47.5mm
The PCB dimensions are the same as the Nextion LCD PCB.
This is so the 2 boards can be stacked separated by nylon spacers.

(Figure 23 a: Top Layer, b: Bottom Layer)

5.3.13 Prototype V3 - Custom PCB V2 Manufacturing

An order for manufacturing of 5 PCB units was placed with JLCPCB of Shenzhen, China with expected turnaround of 5 business days plus delivery of 3-7 business days via DHL.



Screenshot of Order Placement for PCB Manufacturing. (Figure 24)

Feature	Specification
Base Material	FR-4
Layers	4 (Top, 3.3V Power plane, Ground plane, Bottom)
Dimensions	47.5 x 95 mm
PCB Quantity	5
Delivery Format	Single PCB
PCB Thickness	1.6
Impedance	No
PCB Colour	Blue
Silkscreen	White
Surface Finish	HASL (With lead. Production PCBs may be without lead).
Outer Copper Weight	1 oz
Inner Copper Weight	0.5 oz
Material Type	FR-4 Standard Tg 130-140C
Flying Probe Test	Fully Test
Castellated Holes	No
SMT Assembly	None (Assembly will be done manually).

PCB Manufacturing Order Specifications (Table 22)

5.3.14 Lithium Polymer (LiPo) Battery

The documentation for each MKR series board specifies the minimum and maximum capacities for the LiPo battery connected to the board. For the MKR GSM 1400 board a 3.7V 2500mAh capacity battery is recommended as suitable. For the MKR WiFi 1010 board 1024mAh is the minimum recommended capacity. The Custom PCB V2 has footprint space reserved for placement of a LiPo rechargeable battery.

The maximum physical size constraints are: 42mm x 64mm x 9 mm. The 64mm value extends past the edges of the PCB and up to the inner edge of the enclosure.
Design constraints for the PCB layout included ensuring the battery storage surface area on the PCB is as large as possible and clear of any components, solder mask, exposed copper or conductive materials.

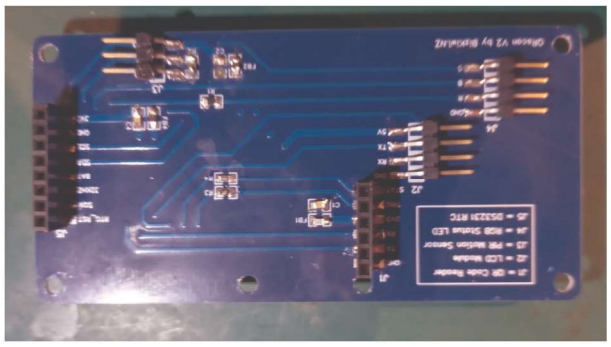
The maximum vertical height of the battery is constrained by the presence of the MKR series board. However, this can be increased by double-stacking the headers which the MKR board plugs into. Note, this will increase the required size of the enclosure.

**LEFT: 3.7V 1200mAh Lithium Polymer (LiPo) Battery used in Prototype V3.
This cell has enough capacity to power a MKR WiFi 1010 MCU.
Dimensions: 33 x 50 x 5 mm.**

**RIGHT: 3.7V 3,030mAh Lithium Ion Prismatic Cell available from Samsung SPI.
This cell has enough capacity to power a MKR GSM 1400 MCU.
Dimensions: 38 x 64 x 9 mm.**

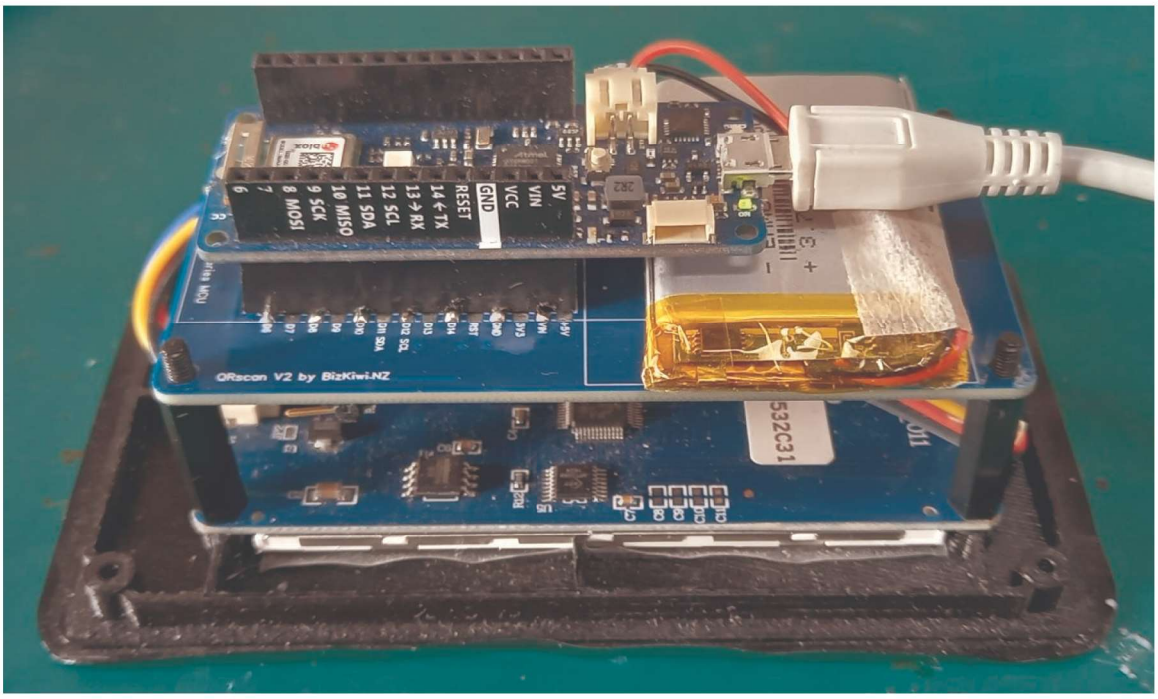
**Both units fit in the reserved PCB and enclosure battery space.
Dimensions: 42 x 64 x 9 mm.
(Figure 25 a, b)**

5.3.15 Prototype V3 - Custom PCB V2 Manufactured and Assembled



Prototype V3 - PCB V2 Manufactured and Assembled with Components Soldered.
Prior to being populated with components the PCBs were tested with a multimeter for circuit continuity by probing solder pads of tracks and pin headers.
a: Testing Bare PCB. b: Testing Assembled PCB. (Figure 26 a,b)

5.3.16 Prototype V3 - Custom PCB V2 and Modules Assembled and Stacked



Prototype V3 - QRscan PCB V2 with LiPo battery stacked beneath a MKR WiFi 1010 MCU and above the Nextion LCD board. (Figure 27)

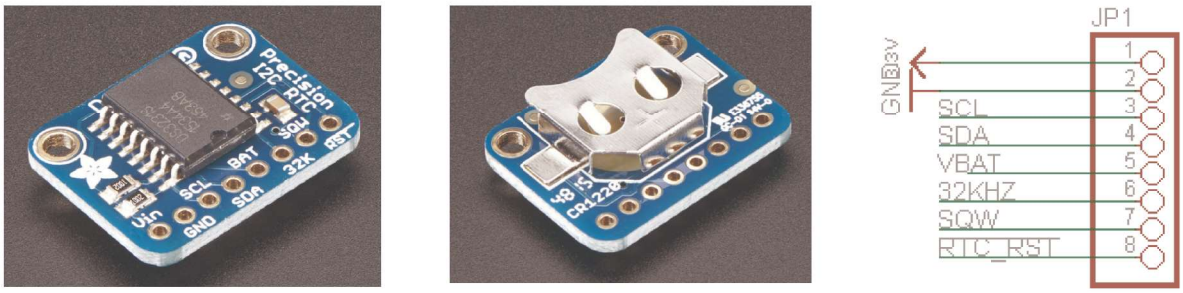
5.3.17 Adafruit DS3231 Real-Time Clock (RTC) Module

Newly added to PCB V2 was a pin header for interfacing with an Adafruit DS3231 Real-time Clock (RTC) module. The DS3231 is one of the most accurate RTC's available with less drift over timespan durations thus enabling more precisely accurate timekeeping on the MCU. The characteristics of clock crystals are such that oscillation is affected by temperature. The Integrated Circuit (IC) of the DS3231 RTC features an embedded temperature sensor situated next to the crystal which enables the IC to apply temperature dependent timing correction offsets to the signal outputs.

Firmware can use the Network Time Protocol (NTP) to synchronise the RTC with known precisely accurate time as kept on Internet-based Network Time Servers.

The MKR GSM 1400 and MKR NB 1500 boards are also able to source current time from the cellular network via the radio module. A third party GPS module could also be used to source UTC time with virtually global coverage.

The DS3231 module features a slot for a battery to provide a backup power source and a 3V CR1220 Lithium ion battery would have a lifespan of up to ~5 years.



Adafruit DS3231 Real-Time Clock (RTC) Module
Dimensions: 23mm x 17.6mm x 7.2mm. Weight: 2.1g. Battery: 3V Lithium CR1220
(Figure 28 a: RTC IC Side, b: Battery Side, c: Module Pinout)

5.3.18 Maintaining Accurate Time on a Microcontroller with RTC and NTP

Many Arduino boards don't feature an accurate Real Time Clock (RTC) and, therefore, their measurement and recording of the passage of time tends to increasingly "drift" the longer the device has been running for. Some RTC's are more precisely accurate than others. A RTC featuring a backup battery power source (e.g. the 3V LiPo CR1220 coin cell battery) can still keep track of the passage of time even in a low power mode or without mains power at all.

Internet connected Microcontroller Units (MCU) have the ability to calibrate the accuracy of their own clocks by comparing their own measurement of time with that of ultra accurate atomic clocks connected to Internet time servers. Atomic clocks measure time in high resolution by measuring the decay of radioactive isotopes. Internet time servers are publicly accessible on the Internet via the Network Time Protocol (NTP). The asynchronous nature of the Internet means that it takes time to transfer data between devices and this can contribute to latency issues when synchronising with a time server. The use of the User Datagram Protocol (UDP) can help to reduce the magnitude of the effects of latency.

Periodic synchronisation of a RTC with an Internet time server via NTP can help to increase general accuracy of time keeping on a MCU without the added hardware costs of a more precisely accurate RTC. Additional methods for obtaining accurate time include interfacing with Global Positioning System (GPS) feeds and cellular networks via an RF module. NTP servers may calibrate from a GPS/GSNN source, a cellular network, a broadcasted RF feed, an on-site atomic clock or a combination of methods.

NTP Provider	Access	Availability	Pool	Server	Support Website
Google	Public	Global	time.google.com	time1.google.com, time2.google.com etc	https://developers.google.com/time
NTP Project	Public	Global	pool.ntp.org	NA	https://www.ntp.org/
Internet NZ	NZ ISPs	Within NZ	Communications.	ntp1.ntp.net.nz, ntp2.ntp.net.nz etc	https://ntp.net.nz/

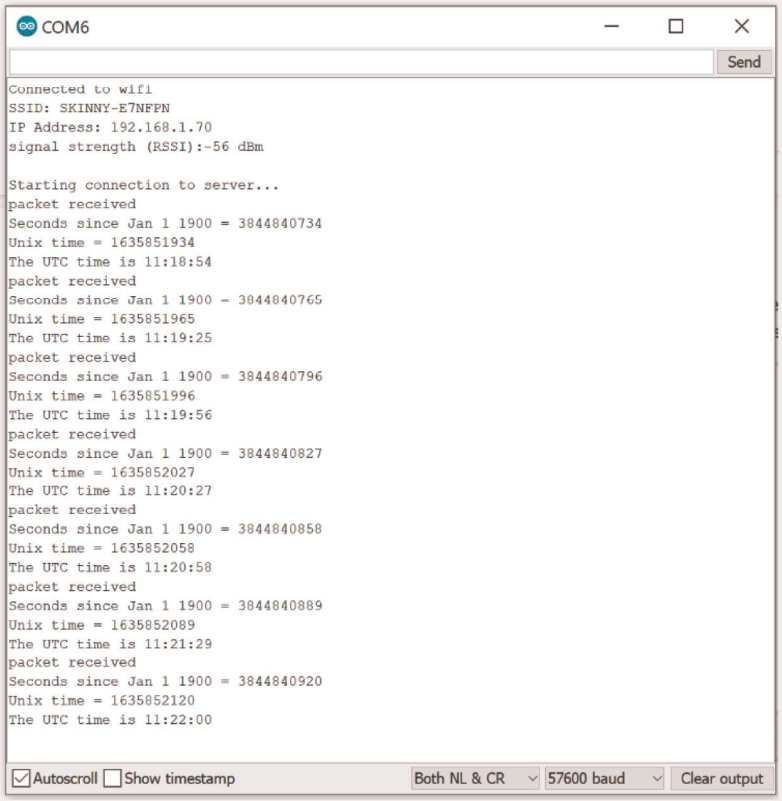
Selection of Network Time Protocol (NTP) Servers (Table 23)

5.3.19 Synchronising the DS3231 RTC with NTP Servers

The firmware was written to periodically synchronise the DS3231 Real-Time Clock (RTC) with a public Network Time Protocol (NTP) server. For simplification the RTC tracks time and date in Universal Coordinated Time (UTC). UTC is used by the NTP servers and is commonly used in web servers and databases. NTP servers apply leap seconds to time keeping as required. Note that the need for leap seconds, although applied in a globally coordinated manner, are not entirely predictable as even an unexpected strong earthquake can slow the rotation of the earth enough for a correction to be needed. UTC does not have daylight savings time which helps to reduce requirements for complicated time and date arithmetic. By applying time zone conversions to UTC at the user interface level a great deal of complexity and potential for error can be avoided.

When testing the firmware it was found that although the code was designed to synchronise the RTC with a NTP server every 30 seconds, in practicality the update was made every 31 seconds. The one second difference is largely due to latency between the MCU and the NTP server and back again. Internet protocols are, for the most part, asynchronous by nature. This means it takes time to shift data between two disparate points. To help minimise this latency, the Network Time Protocol uses the User Datagram Protocol (UDP) which does not use acknowledgement packets to confirm a successful data packet transfer. However, in testing the synchronisation request and response appears to take about 1 second to complete.

To help improve accuracy of the RTC an extra half second could be added to the adjusted time and the interval between synchronisations could be reduced by one second. This would result in more accurate time keeping that takes network latency into account and also correct the update interval to the intended time period.



Synchronising the DS3231 RTC with NTP Servers. (Figure 29)

5.3.20 Applying Timezone and Daylight Savings Offsets to Date and Time

There are some ways to reduce configuration requirements for a device in order to display accurate date and time for the timezone it is located in. Firmware libraries are available for the Arduino that can assist with this. Some of the techniques used include analysis of the IP address and domain the unit has been assigned in order to reveal which geographic locale it has been assigned. If a microcontroller has a GSM interface the cellular network can be polled to determine geographic location as well as local network time. A GPS interface can also be used to determine locale as well as the current UTC time.

For the current project a time offset has been manually assigned to the device to enable date/time arithmetic to be used to determine local time. A more sophisticated solution is certainly possible that would reduce device configuration requirements as well as enable the device to update settings when moved between time zones or Daylight Savings Time (DST) locales.

5.4 Prototype V4 - BearSSL, MQTT, AWS IoT Core, DynamoDB

5.4.1 Prototype V4 Goals

Prototype V2 introduced the PCB V1 Breakout Board (BOB) for modules to interface with and components to be soldered to. Prototype V3 introduced PCB V2 with a pin header for a DS3231 RTC. Connectivity with a NTP Network Time Server was implemented enabling the DS3231 RTC to synchronise time from a highly accurate source. The MKR WiFi 1010 can read time from an NTP server on the Internet via the Nina WiFi radio and the MKR GSM 1400 can source time from the cellular network.

"AWS IoT is a managed cloud platform that lets connected devices - cars, light bulbs, sensor grids, and more - easily and securely interact with cloud applications and other devices." **- AWS**

Primary objectives for Prototype V4 include:

1. No hardware changes in V4. Instead, all changes are related to upgrading and configuration of firmware and AWS cloud infrastructure.
2. TLS and SSL encrypted connectivity with an Amazon Web Services (AWS) IoT Core Cloud server. Scanned QR code data, along with time and date and the device ID can be uploaded as messages to the server via the WiFi Internet connection to enable integration with other online infrastructure.
3. AWS IoT Core needs to be configured with a Service Broker, a Thing (the IoT device), public and private cryptographic keys as encryption Certificates, and an access Policy. Devices connected to AWS IoT Core via the MQTT protocol are required to use x.509 certificates for authentication. For this a public domain example sketch is used to generate a Certificate Signing Request (CSR) on the Arduino board during which a private key is generated, written and permanently locked inside the crypto secure element microchip. The CSR is then manually uploaded to the AWS console to create the x.509 certificate.
4. ArduinoBearSSL is the firmware library to be used for messaging TLS encrypted data between the device and AWS Core IoT.
5. Message Queuing Telemetry Transport (MQTT) is the protocol to be used for sending and receiving messages via connections with the AWS IoT Core server. MQTT is one of the most widely used messaging protocols by IoT devices.
6. Messages uploaded by the device to AWS IoT Core are to be automatically inserted into a custom database table in AWS DynamoDB. This will be enabled by a custom Rule configured in AWS IoT Core. The purpose of this is to enable a robust means of persistent data storage that a future administration application will be able to interface with.

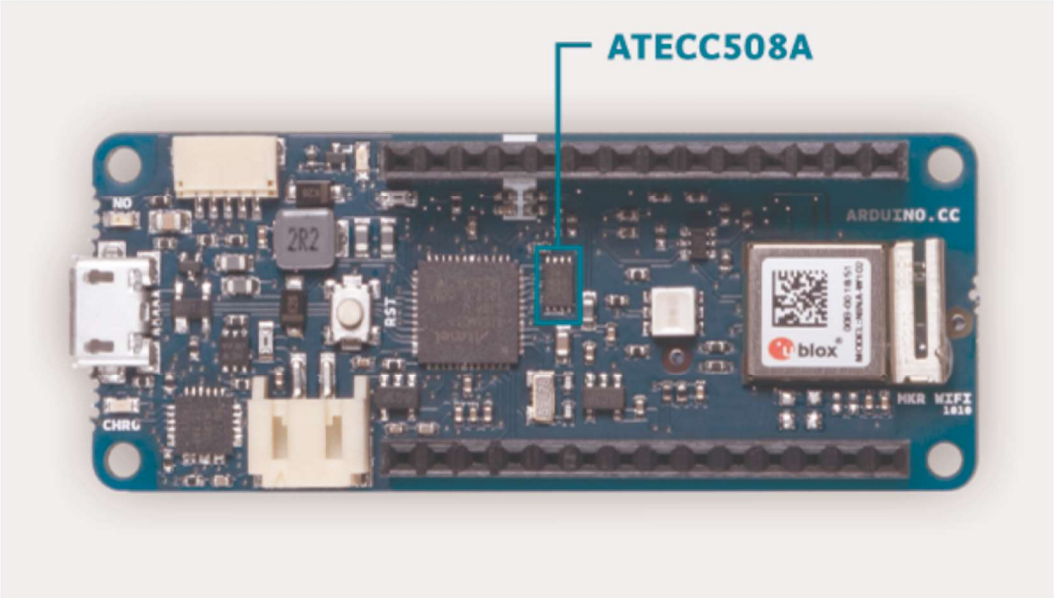
The above objectives for V4 were implemented with the help of a detailed tutorial. See: Arduino_Genuino (16/01/2019)

5.4.2 ArduinoBearSSL

BearSSL is an implementation of the SSL/TLS protocol. **ArduinoBearSSL** is a port from the **BearSSL** library to the Arduino platform and is used for securing MQTT message traffic between an IoT device and an AWS IoT Core server. The ArduinoBearSSL library depends on **ArduinoECCX08** which is a firmware library used for interfacing with the cryptographic key storage functions of the ECCX08 hardware embedded in the MKR series microcontrollers.

5.4.3 ArduinoECCX08 and Secure Element Hardware Cryptography

In order to save memory and improve security, the Arduino.cc development team chose to introduce a hardware secure element to the MKR series and Nano 33 IoT Arduino boards. This crypto hardware is the secure element **ATECC508A** or **ATECC608A** manufactured by Microchip. **ArduinoECCX08** is the firmware library for interfacing with the crypto hardware. The secure element enables offloading part of the cryptography algorithms computational load, as well as to generate, store, lock and manage certificates.



Location of the ATECC508A microchip on the MKR WiFi 1010 MCU (Figure 30)

According to Microchip, the secure elements guarantee two fundamental security properties in communication:

- **Authenticity**: The ability to trust who you are communicating with.
- **Confidentiality**: The ability to be sure the communication is private.

The secure element is used during provisioning to configure the Arduino board for safely connecting with Cloud platform infrastructure. The boards don't use a standard username/password pair credentials authentication. Instead, higher-level authentication, known as client certificate authentication, is implemented.

5.4.4 Message Queueing Telemetry Transport (MQTT) Protocol

According to MQTT.org, MQTT is "The Standard for IoT Messaging".

"MQTT is an OASIS standard messaging protocol for the Internet of Things (IoT). It is designed as an extremely lightweight publish/subscribe messaging transport that is ideal for connecting remote devices with a small code footprint and minimal network bandwidth. MQTT today is used in a wide variety of industries, such as automotive, manufacturing, telecommunications, oil and gas, etc." - MQTT.org.

5.4.5 Prototype V4 Implementation and Configuration

To implement RTC, NTP & MQTT messaging between the MCU and AWS IoT:

1. Generate a Certificate Signing Request (CSR) on MCU using a configuration script.
2. Upload the CSR to AWS IoT Core to generate a certificate.
3. Save the certificate to disk and insert into the MCU configuration firmware.
4. Save the AWS certificate to disk and insert into the MCU configuration firmware.
5. For encryption certificates to work the device needs to keep accurate time and date. Code the firmware sketch to synchronise the DS3231 RTC with a reliable and accurate NTP server at boot time and again periodically to minimise drift effects and maintain high accuracy. The server chosen for this project was a Google public NTP server.
6. The ArduinoBearSSL library is required to be configured in the device firmware.
7. In AWS IoT Core a "Thing" representing the device is required to be configured and with the appropriate certificate and configured access policy attached.

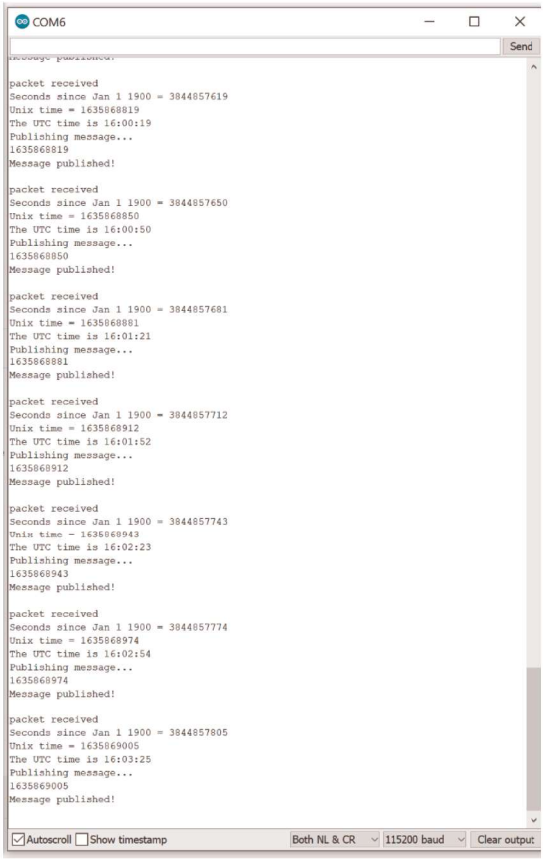
```
-----BEGIN CERTIFICATE-----
HTsH7NvGHOPz2F4JsTo9XN2k6/qWm38hZ8HTsH7NvGHOPz2F4JsTo9XN2k6
U2XkMGU4Sn7wMX8Dz+U2BnHHTsH7NvGHOPz2F4JsTo9XN2k6HqWm38hZ8+W
S4qCH/8HTsH7NvGHOPz2F4JsTo9XN2k64Yx++BnPsWEj5QKn6Z72HdK3XUk
T2BnHHTsH7NvGHOPz2F4JsTo9XN2k6HqWm38hZ8++BnPsWEj5QKn6Z72HdX
HTsH7NvGHOPz2F4JsTo9XN2k6/qWm38hZ8HTsH7NvGHOPz2F4JsTo9XN2k6
U2XkMGU4Sn7wMX8Dz+U2BnHHTsH7NvGHOPz2F4JsTo9XN2k6HqWm38hZ8+W
S4qCH/8HTsH7NvGHOPz2F4JsTo9XN2k64Yx++BnPsWEj5QKn6Z72HdK3XUk
Y3BnHHTsH7NvGHOPz2F4JsTo9XN2k6HqWm38hZ8++BnPsWEj5QKn6Z72HdX
HTsH7NvGHOPz2F4JsTo9XN2k6/qWm38hZ8HTsH7NvGHOPz2F4JsTo9XN2k6
U2XkMGU4Sn7wMX8Dz+U2BnHHTsH7NvGHOPz2F4JsTo9XN2k6HqWm38hZ8+W
S4qCH/8HTsH7NvGHOPz2F4JsTo9XN2k64Yx++BnPsWEj5QKn6Z72HdK3XUk
T2BnHHTsH7NvGHOPz2F4JsTo9XN2k6HqWm38hZ8++BnPsWEj5QKn6Z72HdX
S4qCH/8HTsH7NvGHOPz2F4JsTo9XN2k64Yx++BnPsWEj5QKn6Z72HdK3XUk
P3VnHHTsH7NvGHOPz2F4JsTo9XN2k6HqWm38hZ
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
HTsH7NvGHOPz2F4JsTo9XN2k6/qWm38hZ8HTsH7NvGHOPz2F4JsTo9XN2k6
U2XkMGU4Sn7wMX8Dz+U2BnHHTsH7NvGHOPz2F4JsTo9XN2k6HqWm38hZ8+W
S4qCH/8HTsH7NvGHOPz2F4JsTo9XN2k64Yx++BnPsWEj5QKn6Z72HdK3XUk
T2BnHHTsH7NvGHOPz2F4JsTo9XN2k6HqWm38hZ8++BnPsWEj5QKn6Z72HdX
HTsH7NvGHOPz2F4JsTo9XN2k6/qWm38hZ8HTsH7NvGHOPz2F4JsTo9XN2k6
U2XkMGU4Sn7wMX8Dz+U2BnHHTsH7NvGHOPz2F4JsTo9XN2k6HqWm38hZ8+W
S4qCH/8HTsH7NvGHOPz2F4JsTo9XN2k64Yx++BnPsWEj5QKn6Z72HdK3XUk
Y3BnHHTsH7NvGHOPz2F4JsTo9XN2k6HqWm38hZ8++BnPsWEj5QKn6Z72HdX
HTsH7NvGHOPz2F4JsTo9XN2k6/qWm38hZ8HTsH7NvGHOPz2F4JsTo9XN2k6
U2XkMGU4Sn7wMX8Dz+U2BnHHTsH7NvGHOPz2F4JsTo9XN2k6HqWm38hZ8+W
z2F4JsTo9XN2k6/qWm38hZ8HuX3H7vYd
-----END CERTIFICATE-----
```

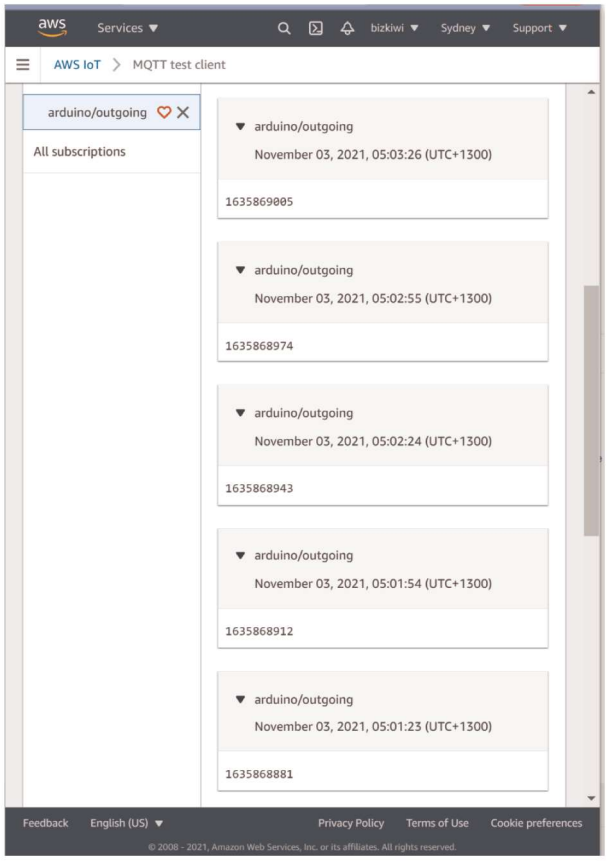
Prototype V4 - Format of Device Encryption Certificates (Figure 31)

The top (fake) certificate is for the Device.
The bottom (fake) certificate is the public key for the AWS server.

5.4.6 Functional Testing of Server Messaging

Functional testing of messaging between the device and AWS IoT Core was mostly successful. However, it was discovered that when two successive message events occur very close together occasionally the latter message would arrive at the server before the earlier message. This advocates a resolution by time stamping messages at the device rather than after they are received at the server so as to enable correct time sorted ordering of messages when they are to be inserted into a database table.





Prototype V4 - Testing (Figure 32 a,b)

ABOVE LEFT: Serial monitor event messages output from the Arduino USB port. Newest messages are at the bottom of the window.

ABOVE RIGHT: Incoming MQTT messages from the Arduino in AWS IoT Core user interface. Newest messages are at the top of the window.

5.4.7 Insertion of Messages into AWS DynamoDB

AWS DynamoDB is a highly scalable and robust NoSQL database. For Prototype V4 a database table was created in AWS DynamoDB and designed to accept the name/value data pairs uploaded by the device to AWS IoT Core. AWS IoT Core does not store messages historically and so DynamoDB takes on the role of enabling highly robust persistent message storage in the AWS Cloud.

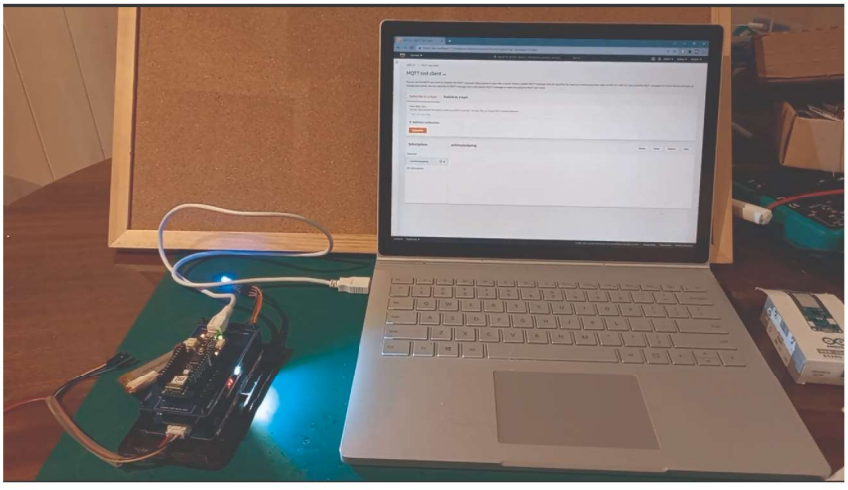
5.4.8 Device Status and RGB LED Output Values

Status		RGB LED		PWM Pins Output Value		
Code	Description	Colour	Flashing	Red	Green	Blue
0	Waiting for the server.	Red	✗	255	0	0
1	Server error.	Red	✓	255	0	0
2	Systems Nominal.	Green	✗	0	255	0
3	Data is in transit.	Green	✓	0	255	0
4	Message sent.	Blue	✗	0	0	255
5	New message received.	Blue	✓	0	0	255
6	Reserved.	Orange	✗	128	128	0
7	Reserved.	Orange	✓	128	128	0

Device Status and their Corresponding RGB LED Output Values (Table 24)
The Flashing ON and OFF cycle has a frequency of 1Hz.

5.4.10 Prototype V4 Demonstration

Features and functionality added to firmware since Prototype V3 include AWS IoT Core secure messaging enabled by the secure element microchip and TLS encryption features of the **ArduinoBearSSL**, **ArduinoECCX08** and **MQTT** firmware libraries.



Prototype V4 Demonstration. (Figure 33)
Complete Firmware Code for Prototype V4 is included in **Appendix B.**

5.5 Prototype V5 - PCB V3, LCD GUI, Touchscreen, Enclosure

5.5.1 Looking Ahead

While Prototype V5 has not been fully implemented, some planning and preparation has already taken place which will be covered in the next few pages.

5.5.2 Prototype V5 Goals

Prototype V2 introduced a PCB Breakout Board (BOB) for modules to interface with and components to be soldered to. The LCD module was connected and tested via programmatically writing some very simple output to the display via the serial data link from the Arduino MKR WiFi 1010 microcontroller. However, a Graphical User Interface (GUI) is still undeveloped.

Primary objectives for Prototype V5 include:

1. Produce a revised design iteration of the component adapter board (PCB V3) to include circuit design optimisation and repositioning of components and headers for more efficient use of available space on the PCB and future enclosure.
2. Display custom status messages on the LCD display triggered by PIR motion sensor activation and QR code scanning.

Further stretch goals include:

1. Design, implement and test an interactive touchscreen GUI for the LCD. This can be for the user to customise options for collecting their loyalty reward.
2. Design and 3D print an enclosure to protect and house the assembled device.
3. Larger PCB surface area.
4. Larger battery storage surface area enables larger capacity LiPo battery.
5. DC-IN 5.5mm Barrel connector
6. DC-DC buck voltage regulator (up to 30V IN, regulated 5V @ <3A OUT)
7. In-line self resetting power supply fuse on voltage regulator output (6V/3A)
8. Remove redundant micro-USB power cable.
9. Remove cabling for connecting RGB LED and PIR motion sensor as they now instead interface directly with SMD female pin headers.
10. Add to PCB a DHT-12 Thermistor (Temperature and Humidity Sensor) for the purpose of proactively monitoring the internal environment of the enclosure.
11. Add to PCB a 9-pin female header for interfacing an NFC read/write module.

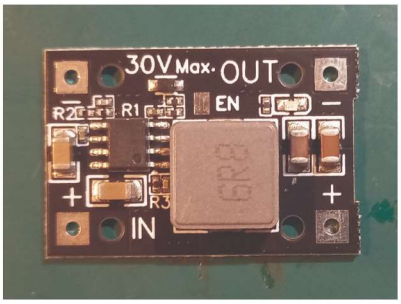
5.5.3 Components and Power Budget

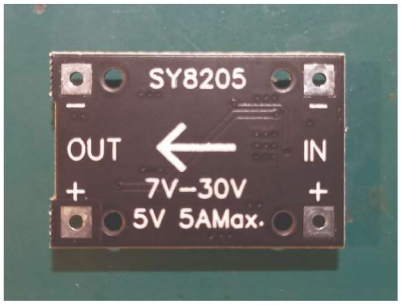
Part Description	Function	Power (mA)	Quantity
Arduino MKR WiFi 1010 (3.3V, BLE, WiFi)	Microcontroller.	<20	1
BLE module.	Communications (can be disabled).	<=47	NC
WiFi module.	Communications (can be disabled).	<=112	NC
DS3231 Real-Time Clock (RTC) module	**Time and Date tracking.**		**1**
3V CR1220 Lithium Coin Battery for RTC	**Time and Date tracking.**	**NA**	**1**
3.7V 1200mAh LiPo Battery	**Rechargeable Battery.**	**NA**	**1**
Barcode/QR code reader module	Barcode and QR code scanning	<190	1
Liquid Crystal Display (LCD) Module	**Display of user instructions, status**	**<85**	**1**
RGB LED indicator	Device status notification	<18	1
AM312 Passive Infrared Detector (PIR)	Human proximity detection	<20	1
Component interface adapter PCB	**Interfacing of device components**	**0**	**1**
SMD Series Resistors	Current limiting to protect IO pins	<5	3
SMD Pull Up Resistors	Ties logic high instead of floating.	0	0
SMD Pull Down Resistors	Ties logic low instead of floating.	<1	1
SMD Ceramic Capacitors	**Decoupling / Bypass Capacitors**		4
SMD Ferrite Bead Power Pin Filters	**Protect from high frequency noise**		4
3D Printed Front Panel	**Bezel for LCD Module**		
Enclosure	**Housing of Device Componentry**	**0**	**1**
DC-DC Buck Voltage Regulator	**<30V IN, Regulated 5V @ <3A OUT)**	**?**	**1**
DHT-12 Thermistor	**Temperature and Humidity Sensor**	**?**	**1**
NFC RFID Read/Write Module	**Tagging, Identity, Pairing, Events**	**?**	**1**
SMD Piezoelectric Buzzer	**Sonic Notifications of Events**	**?**	**1**
SMD Resistor in series with Buzzer	**330ohm between Buzzer and GND**	**?**	**1**
Contingency		**>100**	**mA**
	Total Power Budget	**~598**	**mA**

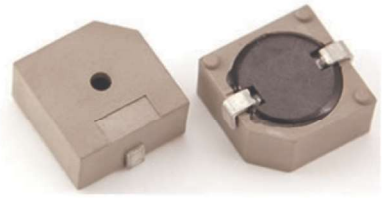
Prototype V5 - Parts and Their Functions (Table 25)

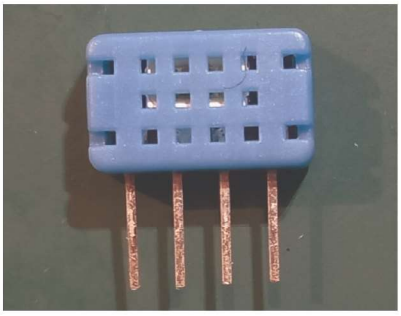
GREEN coloured text refers to components added in Prototype V3.
RED coloured text refers to components added in Prototype V5.
Note: WiFi and BLE normally can't be used simultaneously unless the user has hacked the device. Hacking the device in this way will void the warranty.

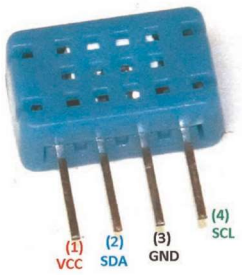
5.5.4 New Components for PCB V3 (Prototype V5)











Prototype V5 - New Components to be added to PCB V3 Rev.2.

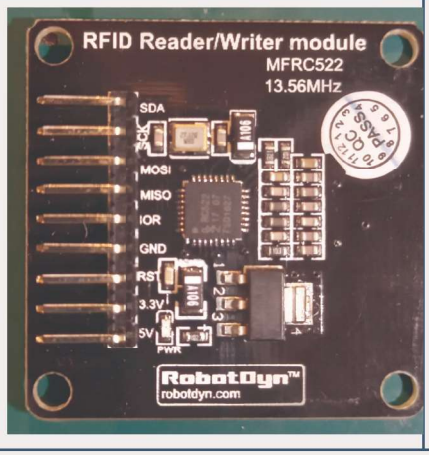
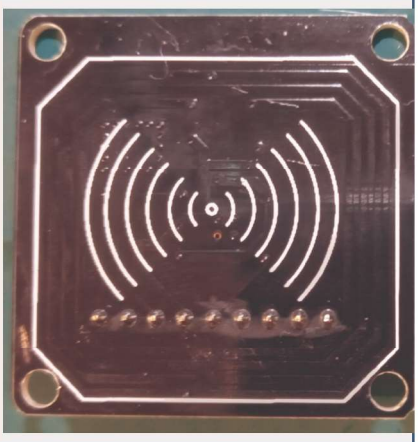
TOP LEFT, MIDDLE: DC-DC Buck Voltage Regulator (IN up to 30V, OUT 5V up to 5A). A 6V/5A self-resetting fuse will be included inline with the output.

TOP RIGHT: SMD Piezoelectric Buzzer (Active HIGH). A 330ohm resistor will be included inline between the Buzzer and GND.

BOTTOM LEFT, MIDDLE: DHT-12 Thermistor Temperature and Humidity Sensor for proactively monitoring the enclosure environment. I2C Pinout is shown. Precision for the DHT-12 readings is in half degree increments.

(Figure 34 a, b, c, d, e)

5.5.5 NFC Module J6 Header Pinout

Pin	Pin Name	Pin Used	NFC Module Front	NFC Module Back
0	SDA	✅		
1	SCK/SCL	✅		
2	MOSI	❌		
3	MISO	❌		
4	IOR	❌		
5	GND	✅		
6	RST	❌		
7	3.3V	✅		
8	5V	❌		

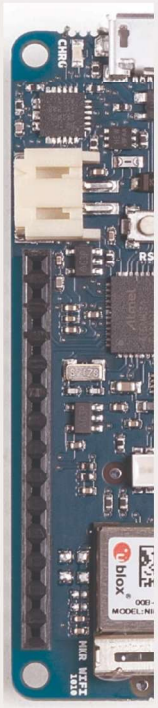
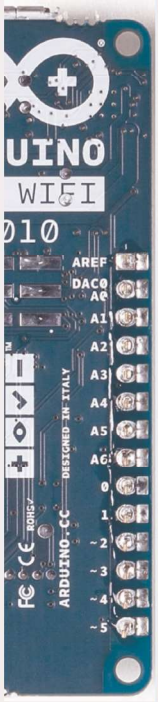
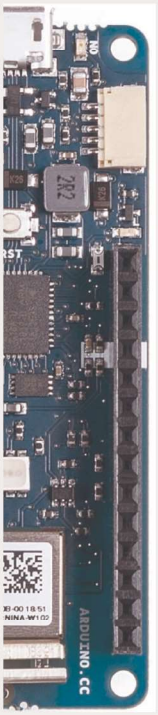
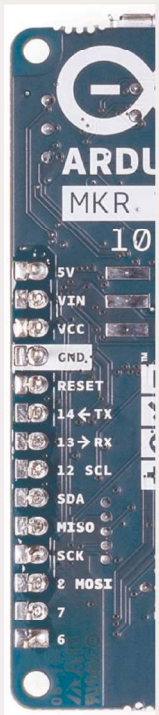
Near Field Communications (NFC) MFRC522 Module Header Pinout (Table 26)
The module supports SPI, UART and I2C interfaces.
We will use I2C to reduce the number of MCU pins that are used.

5.5.6 Prototype V5 - Custom PCB V3 Pin Headers

Designator	Module / Component	Interface	Header Pins	Footprint	PCB Side
H1	Arduino - Left side pin header	NA	15	SMD 2.54mm	Bottom
H2	Arduino - Right side pin header.	NA	15	SMD 2.54mm	Bottom
J1	QR Code Reader Module	UART	6	SMD 2.54mm	Top
J2	LCD Module	UART	4	SMD 2.54mm	Top
J3	PIR Module	Digital IN	3	SMD 2.54mm	Top
J4	RGB LED	3 x PWM	4	SMD 2.54mm	Top
J5	DS3231 Real-time Clock (RTC)	I2C	8	SMD 2.54mm	Top
J6	NFC RFID Read-Write Module	I2C	9	SMD 2.54mm	Bottom
J7	DHT-12 Thermistor	I2C	4	SMD 2.54mm	Bottom

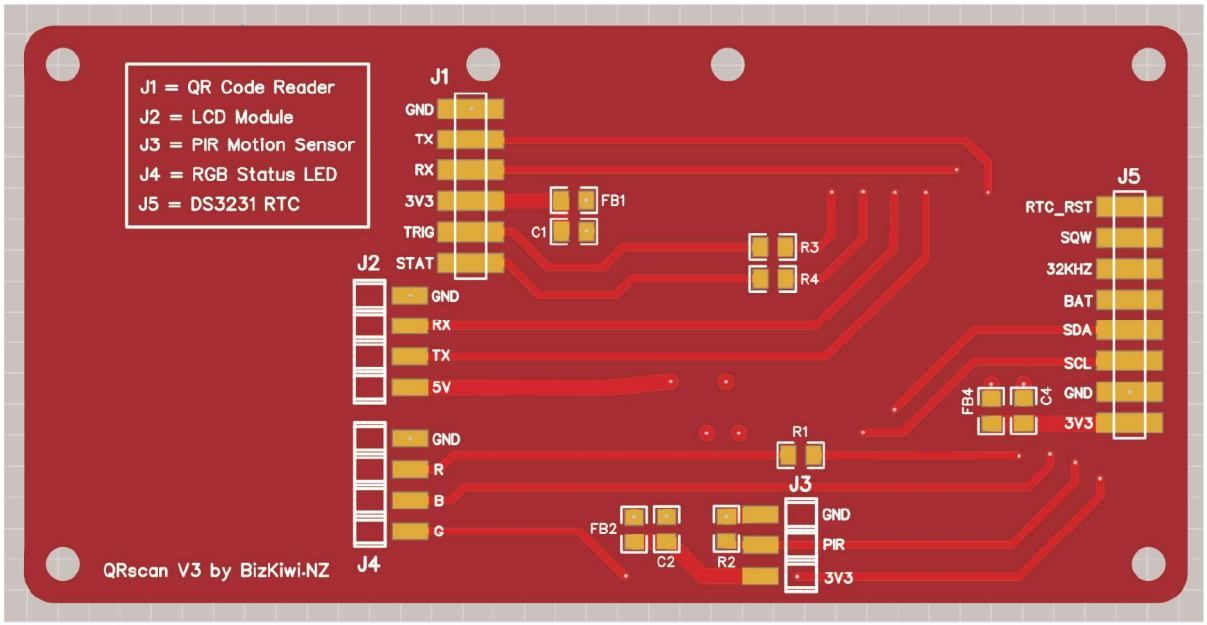
PCB V2 Pin Headers (Table 27)
GREEN = new addition to Prototype V3
RED = new addition to Prototype V5

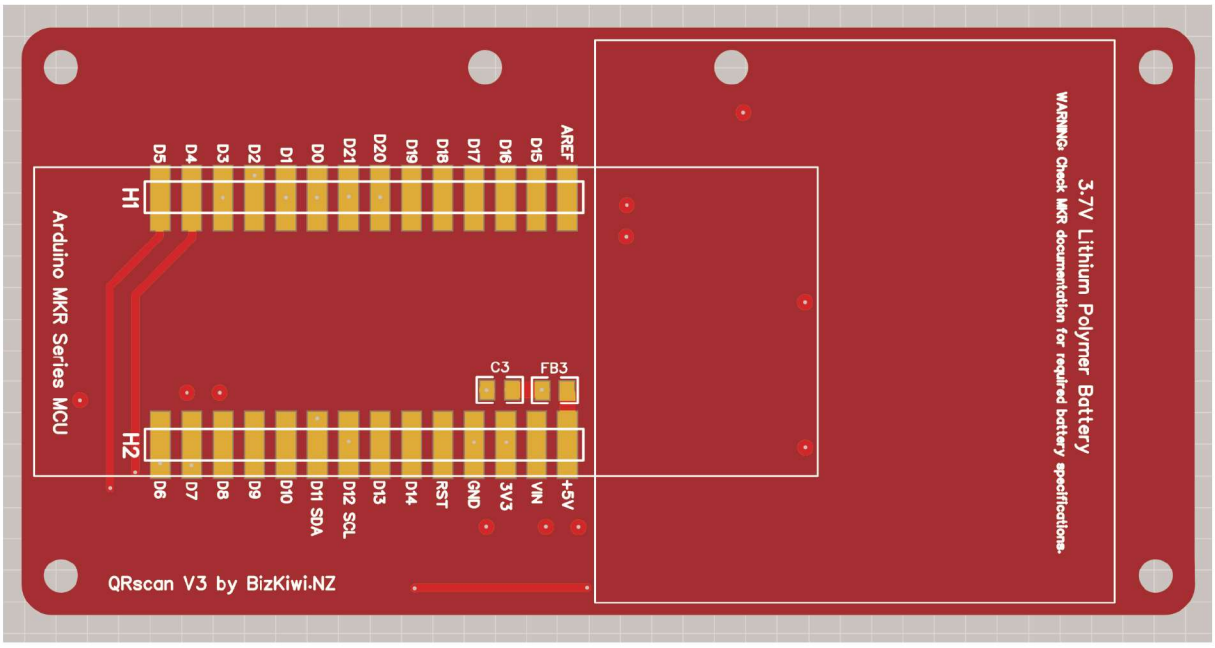
5.5.7 Prototype V5 (PCB V3) - Arduino MKR WiFi 1010 Pinout Allocations

Arduino PCB			Arduino MKR WiFi 1010	Connected Module / Component		
PCB Top	PCB Bottom	Pin Name	Config	Component	Pin Name	Config
						
		AREF	Reserved for Shields			
		DI5 AO DACO	Reserved for Shields			
		DI6 AI	Reserved for Shields			
		DI7 A2	Output, NFC RST	NFCMODULE	RST	Input
		~DI8 A3	Reserved for Shields			
		~DI9 A4	Output, PiezoBuzzer	BUZZER	BUZZ+	Input
		D20 A5	Output, TriggerToQRScan	QRSCAN	TRIG	Input
		D2I A6	Input, StatusFromQRScan	QRSCAN	STAT	Output
		~DO	SERCOM3	NEXTIONLCD	RX	RX Input
		~DI	SERCOM3	NEXTIONLCD	TX	TX Output
		~D2	SERCOMO	QRSCAN	RX	RX Input
		~D3	SERCOMO	QRSCAN	TX	TX Output
		~D4	Input, SenseFromPIR	PIR	PIR	Output
		~D5	PWM Output	RGBLED	GREEN	Anode
PCB Top	PCB Bottom	Pin Name	Config	Component	Pin Name	Config
						
		5V	NEXTIONLCD			
		VIN	Voltage Regulator Output			
		VCC (3V3)	QRSCAN, PIR, RTC, NFCMODULE, DHTI2			
		GND	NEXTIONLCD, QRSCAN, PIR, RGBLED, RTC, NFCMODULE, BUZZ+330R, DHTI2			
		RESET				
		DI4 TX	SERCOM5 (Reserved)	SPARE UART	RX	RX Input
		DI3 RX	SERCOM5 (Reserved)	SPARE UART	TX	TX Output
		~DI2 SCL	I2C Bus	DS3231 RTC, DHTI2		
		DII SDA	I2C Bus	DS3231 RTC, NFCMODULE, DHTI2		
		~DIO MISO	Reserved for Shields			
		D9 SCK	Clock	NFCMODULE		
		~D8 MOSI	Reserved for Shields			
		~D7	PWM Output	RGBLED	RED	Anode
		~D6	PWM Output	RGBLED	BLUE	Anode
KEY: SO = Serial (USB/HID), SI = Serial1 (QRSCAN), S2 = Serial2 (NEXTIONLCD)						

Prototype V5 (PCB V3) - Arduino MKR WiFi 1010 Pinout Allocations (Table 28)

5.5.8 Prototype V5 - Custom PCB V3 Rendered Design





Prototype V5 - Design of Custom PCB V3 as Rendered in EasyEDA.

While this PCB iteration is functionally identical to PCB V2, improvements have been made such as optimisation of track layouts and central positioning of the J4 pin header in order to reduce mechanical stress placed on connected cables.

A Revision version would add further components described in 5.5.4 and 5.5.5.

PCB Dimensions: 95mm x 47.5mm

The PCB dimensions are the same as the Nextion LCD PCB.

This is so the 2 boards can be stacked separated by nylon spacers.

(Figure 35 a: Top Layer, b: Bottom Layer)

5.5.9 Nextion Touchscreen LCD Display Specifications

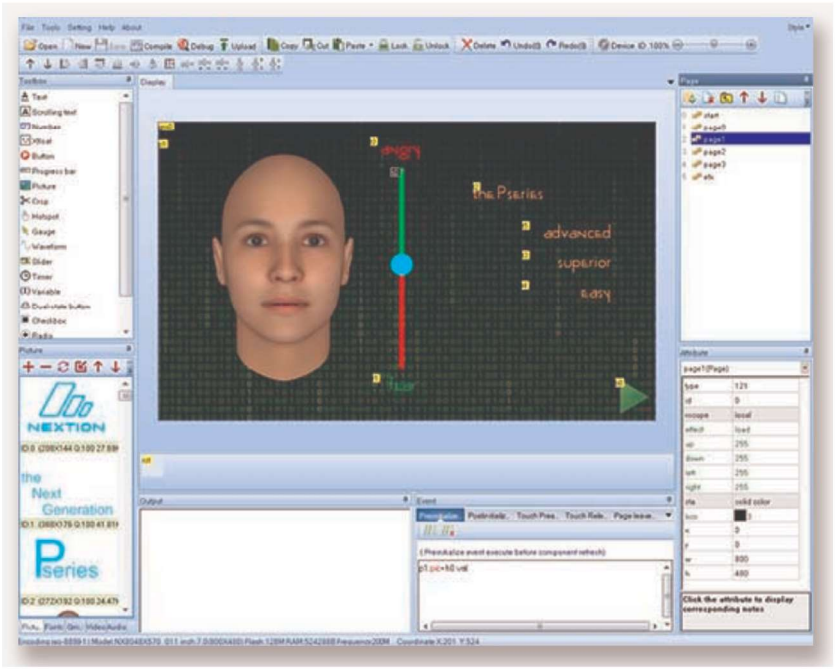
"Nextion is a seamless Human Machine Interface (HMI) solution that provides a control and visualization interface between a human and a process, machine, application or appliance. Nextion is mainly applied to (the) Internet of Things (IoT) or consumer electronics fields. It is the best solution to replace the traditional LCD and LED Nixie tube." - ITEAD (2021).

Nextion 3.2" Touchscreen LCD Module (Figure 36)

Feature / Function	Specification
Model Name	Nextion 3.2" Basic Series HMI Touch Display
Model Number	NX4024T032
Manufacturer	ITEAD Intelligent Systems Co. Ltd
Display Type	TFT Screen with integrated 4-wire Resistive Touch Panel
Display Size	3.2"
PCB Dimensions	95 x 47.5mm
Resolution	400 x 240 px
Visual Area	69.60mm (L) x 41.6mm (W)
Colour Range	RGB 65K true to life colours
Adjustable Brightness	0-230 nit, the interval of adjustment is 1%
Touch Panel	Resistive Touch Panel (RTP)
Interface	Easy 4 pin interface to any TTL Serial Host
USART Port	XH2.54 4Pin
Module MCU	48 MHz
Flash Memory	4 MB (for User Application Code and Data)
SRAM	3584 Byte
Firmware Upgrading	On board micro-SD Card (FAT32 format)
Input Power	DC 5V @ 500mA
Power Consumption	DC 5V @ 85mA
Certifications	CE/EMC, RoHS certified (certificates)

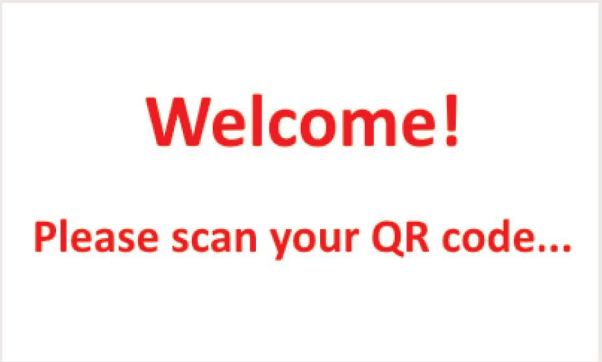
Nextion Touchscreen LCD Display Specifications (Table 29)

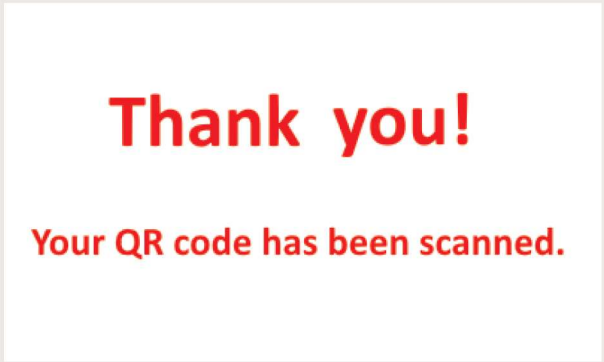
5.5.10 Touchscreen Interface Editor Software



Nextion Editor Software Graphical User Interface (GUI) for Design of Touchscreen User Interface. (Figure 37)

5.5.11 Simple Prompt and Status Messages for the LCD Module





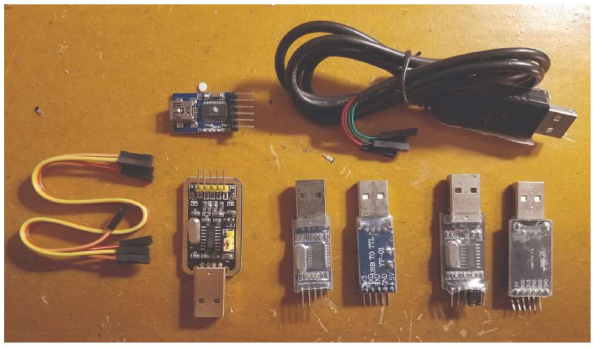
Simple Prompt and Status Messages for the LCD Module (Figure 38 a, b)

5.5.12 Uploading Touchscreen User Interface Software and Media to LCD

The three methods for uploading UI software to LCD Touchscreen are:

1. Serial connection from Arduino microcontroller.
2. Load software on SD card on LCD controller PCB.
3. USB to TTL converter (to upload software directly from PC).

Uploading via the USB to TTL converter is the fastest method for updating the LCD controller with changes to the GUI.



Uploading Graphical User Interface (GUI) Software to Touchscreen (Figure 39 a, b)

USB to TTL converter connected to serial lines of LCD controller PCB.

5.5.13 Enclosure Design Constraints

If a barcode or QR code is too close to the camera it may not scan correctly.
The space under the camera needs to be big enough for codes to scan correctly.
A minimum height of 50mm would enable most barcode formats to scan without errors. The actual height used in the enclosure design was 80mm.

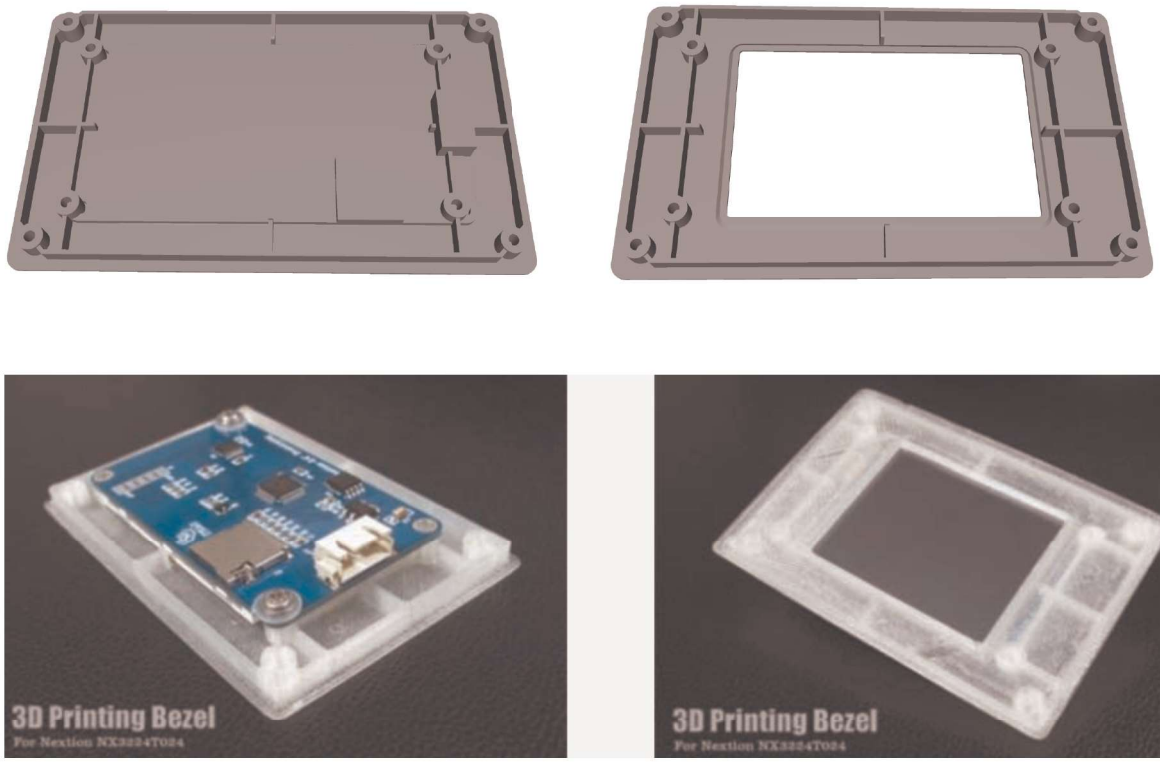
Barcode	Precision	Near	Far
Code 39	0.1mm(4mil)	30mm	70mm
Code 39	0.127mm(5mil)	25mm	100mm
Code 39	0.5mm(20mil)	45mm	250mm
Code 39	1mm(40mil)	110mm	400mm
EAN-13	0.33mm(13mil)	50mm	200mm
QR Code	20.0mil	30mm	210mm
Data Matrix	10mil	45mm	75mm
PDF 417	6.8mil	25mm	100mm

Test Conditions: T=25℃, Illumination=200LUX, PCS=0.9

How the Scanner Module Depth of Field affects Design of Enclosure (Table 30)

Source: QR Code Scanning Module Datasheet

5.5.14 Enclosure Design Inspiration



3D printable parts, including Bezel and Base panels, for the LCD Module. These parts were made available for download by the manufacturer of the Nextion LCD module. (Figure 40 a,b,c,d)

5.5.15 3D Printing of Enclosure

3D Printer used to print the Bezel for the Device Enclosure (Figure 41)

6. Manufacturing Budget

6.1 Prototype V1

6.1.1 Bill of Materials (BOM)

Total materials costs were $123.05. The BOM for Prototype V1 is in **Appendix F.**

6.2 Prototype V2

Prototype V2 was never manufactured as work instead proceeded to Prototype V3.

6.3 Prototype V3

6.3.1 Bill of Materials (BOM)

Total materials costs were $231.55. The BOM for Prototype V3 is in **Appendix F.**

6.3.2 PCB Manufacturing for Prototype V3

Supplier	Budget Items (in-house labour excluded)	QTY	NZD/Unit	Sub-Totals
JCLPCB	Custom Adapter PCB (from Gerber files)	5	1.60	8.00
DHL	Shipping (DHL from China to New Zealand)	1	32.00	32.00
			TOTAL (ex GST) $	**40.00**

Prototype V3 - PCB V2 Manufacturing (Table 31)

6.4 Prototype V4

Prototype V4 was never manufactured because the only new changes were in firmware and Cloud configuration. No new hardware changes were in Prototype V4.

6.5 Prototype V5

Although some planning and design had taken place for Prototype V5 no practical implementation work had taken place before the overall project deadline.

6.6 Budget Summaries with Contingency Funding

6.6.1 Budget Summary for Prototype V1

Budget Items (in-house labour excluded)	Amounts
Bill of Materials (BOM)	123.05
Sub-Total $	**123.05**
Contingency (20%)	24.61
TOTAL $ (ex GST)	**147.66**

Prototype V1 - Budget Summary (Table 32)

6.4.2 Budget Summary for Prototype V2

All new changes in Prototype V2 were in design and firmware.
There were no manufacturing costs as work had proceeded directly to V3.

6.4.3 Budget Summary for Prototype V3

Budget Items (in-house labour excluded)	Amounts
Bill of Materials (BOM)	191.55
PCB Manufacturing (V2)	40.00
Sub-Total $	**231.55**
Contingency (20%)	46.31
TOTAL $ (ex GST)	**277.86**

Prototype V3 - Budget Summary (Table 33)

6.4.4 Budget Summary for Prototype V4

There were no hardware changes or manufacturing costs introduced with V4.
This is because all changes in V4 were in firmware and server systems.

6.4.5 Budget Summary for Prototype V5

No hardware changes or manufacturing costs were implemented in V5 before the overall project concluded. Additional components for the PCB V3 Revision 2 would include those described in **5.5.4 and 5.5.5.**

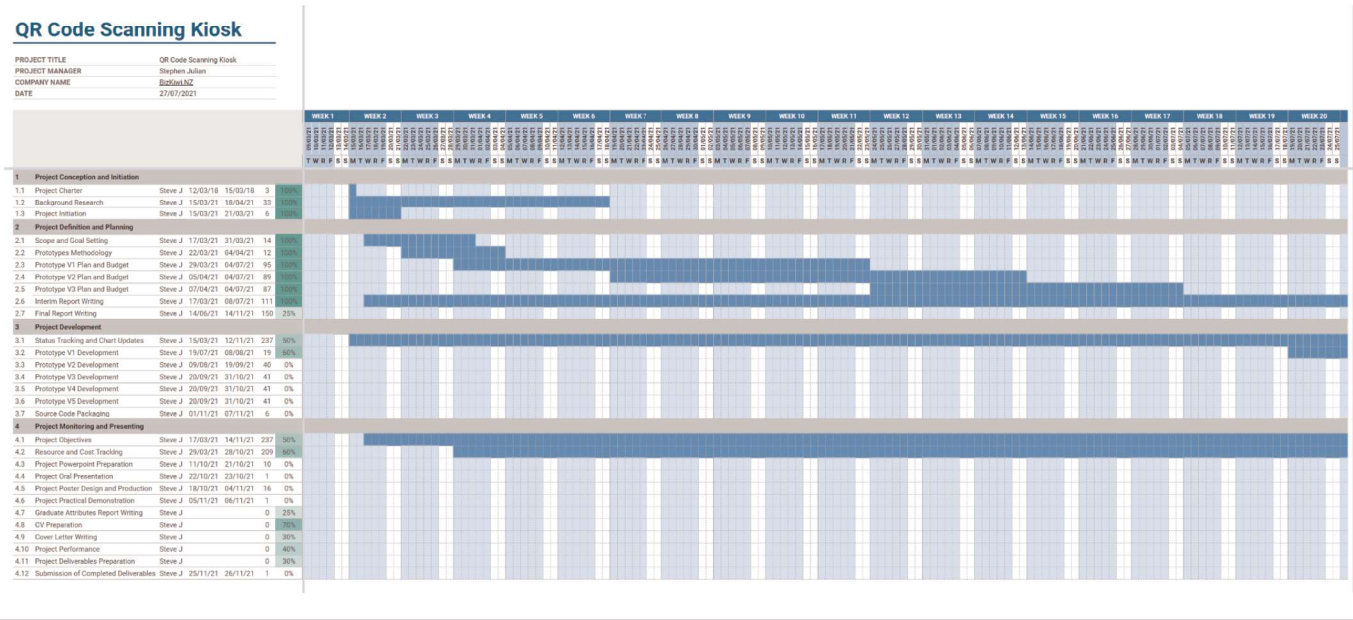
6.4.6 Budget Summary for Overall Project

Budget Items (in-house labour excluded)	Amounts
Summary for Prototype V1	147.66
Summary for Prototype V3	277.86
TOTAL $ (ex GST)	**425.52**

Overall Project - Budget Summary (Table 34)

7. Timeline

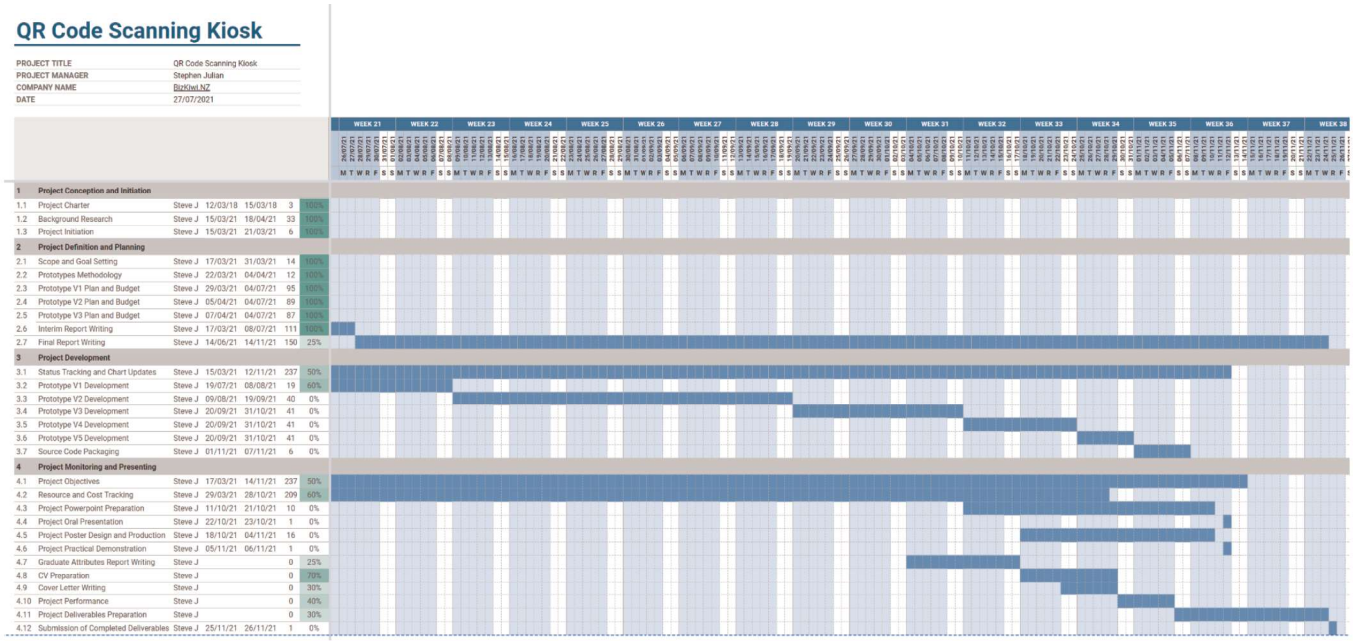
7.1 Timeline - Semester A (Research Phase)



Gantt Chart Interim Timeline - Semester A (Research Phase) (Chart 2)

This Gantt Chart is displayed in full in **Appendix G.**

7.2 Timeline - Semester B (Practical Phase)



Gantt Chart Projected Timeline - Semester B (Practical Phase) (Chart 3)

This Gantt Chart is displayed in full in **Appendix G.**

8. Conclusions

8.1 Prototypes Status and Outcomes

Since the **Interim Report** was completed a number of changes were made to the number of prototype versions and the design scope for each prototype version. These changes, including new features and functionality, were made based on new information and research findings since the completion of the **Interim Report**.

Version	New Functions, Features, Changes	Status and Outcomes
Prototype V1	Breadboard System, Scanner, PIR, RGB LED.	Successfully tested and demonstrated.
Prototype V2	Add PCB V1, LCD Display. Configure SERCOM in firmware to configure and enable multiple hardware serial ports for LCD and USB.	Prototype V2 PCB V1 was not manufactured as correctly operating SERCOM configuration and operation was unable to be attained with breadboard versions of Prototype V1 and V2. Work instead progressed to Prototype V3 and PCB V2.
Prototype V3	Swap for MKR MCU, Update to PCB V2, Add SERCOM configuration, LiPo Battery, DS3231 RTC, WiFi, add NTP support to firmware. Manufacture PCB V2.	For Prototype V3 PCB V2 was manufactured and assembled by JLCPCB in Shenzhen and, following successful circuit continuity testing, a PCB was assembled in-house ready for operational testing. All work was completed and tested successfully.
Prototype V4	Add BearSSL, ArduinoECCX08, MQTT firmware libraries for implementing secure messaging between the device and AWS IoT Core. Configure AWS IoT Core with Rules for inserting uploaded messages into the AWS DynamoDB NoSQL database.	No hardware changes were introduced with Prototype V4. All new changes were in firmware and configuration of AWS IoT Core and AWS DynamoDB services. All work was completed and tested successfully.
Prototype V5	Add Touchscreen LCD Graphical User Interface (GUI), Design and 3D print an Enclosure. Update to PCB V3 for design optimization improvements. Update to PCB V3, add Piezoelectric Buzzer (Active HIGH). Add pin header for NFC Read/Write Module. Add DHT-12 thermistor temperature and humidity sensor for proactive monitoring of the enclosure environment. Add underneath the battery storage area the DC-DC buck voltage regulator (up to 30V IN, 5V/5A OUT), DC-IN barrel connector and self resetting 6V/5A inline fuse. Remove redundant USB power cable.	Design and planning initiated. Implementation work remains as future opportunities for improvement.
Beyond V5	For PCB V4 strictly choose SMD components and footprints stocked by JLCPCB to enable auto pick and place assembly of components when the PCBs are ordered for manufacturing. Lengthen PCB V4 by ~15mm to enable soldering of the RGB LED and PIR module headers directly to the PCB (no wires).	Conceptual stage for establishing new prototype objectives as opportunities for improvement.

Status and Outcomes of Prototypes Testing and Implementation (Table 35)

8.2 Opportunities for Improvement

8.2.1 Reduce Number of Costly Parts

Module	Microcontroller	Function / Purpose
Arduino MKR WiFi 1010	SAMD21 Cortex®-M0+ 32bit LE ARM MCU	Arduino Controller
Arduino MKR WiFi 1010	NINA-W10?	Communications
QR Code Module	?	Camera Controller
Nextion LCD Module	48MHz MCU	LCD Controller / GPU

Quantity of Microcontrollers used in Prototype V3 (Table 36)

8.2.2 Potential Design Improvements

Connectivity Options	Application / Purpose / Benefit
Upgrade Touchscreen	Upgrade the resistive touchscreen display to a capacitive model to improve reliability, consistency and user friendliness
Add NFC Tag Scanning Module	Add an NFC scanner to provide the customer with options for scanning their phone the same way as smartphone users can use a mobile app for pay-and-tap.
Improve Battery Heat Dissipation	Update the PCB to include an array of large diameter via holes in the battery storage area to assist with cooling the LiPo battery.
Additional Connectivity Options	See next table below.
Single Board PCB	All components and modules on a single integrated PCB.
Suitable Enclosure	Strong, water resistant and space saving enclosure.

Potential Future Design Improvements (Table 37)

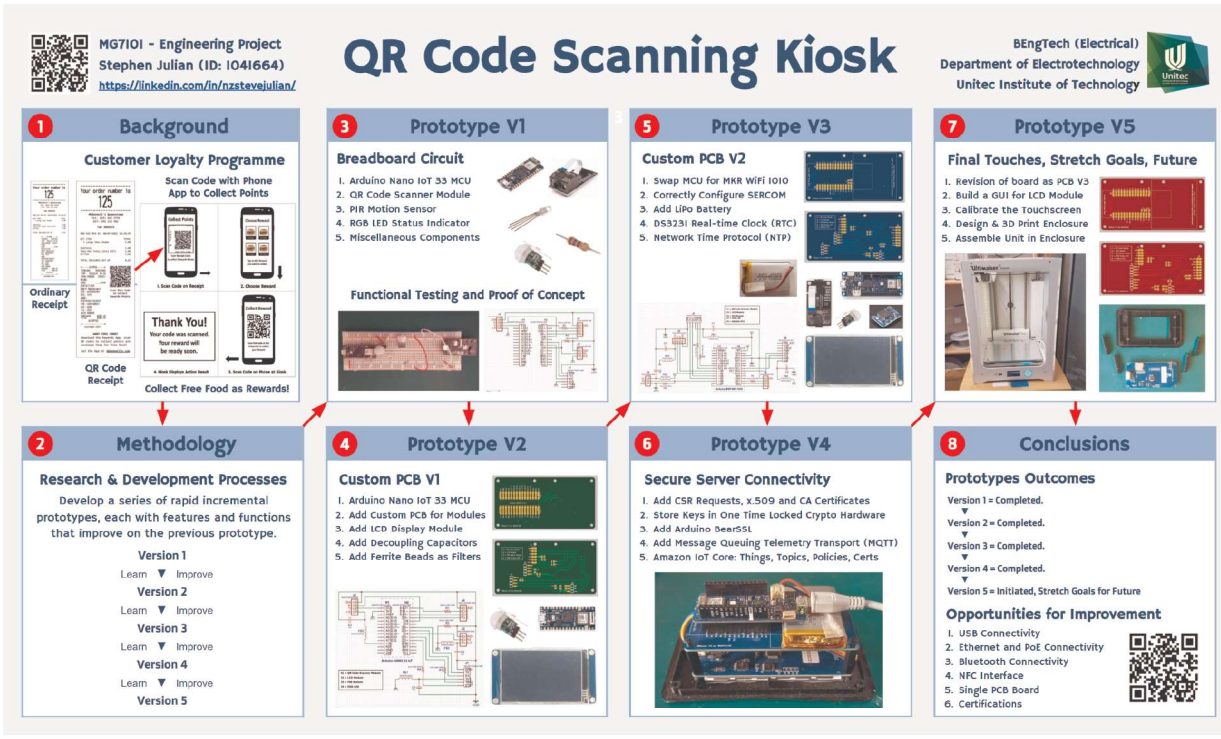
8.2.3 Potential Connectivity Options

Connectivity Options	Application / Purpose / Benefit
USB	Localised data transmission for added security.
Ethernet	Faster and more reliable connectivity.
Power-over-Ethernet (PoE)	Single cable for power and data connection to server.
Bluetooth connectivity	Emulation of bluetooth keyboard.

Potential Future Connectivity Options (Table 38)

8.3 Project Poster

To see a larger version of the Project Poster Design refer to **Appendix I.**



Project Poster Design (Figure 42)

A larger depiction of this poster is included in **Appendix I.**

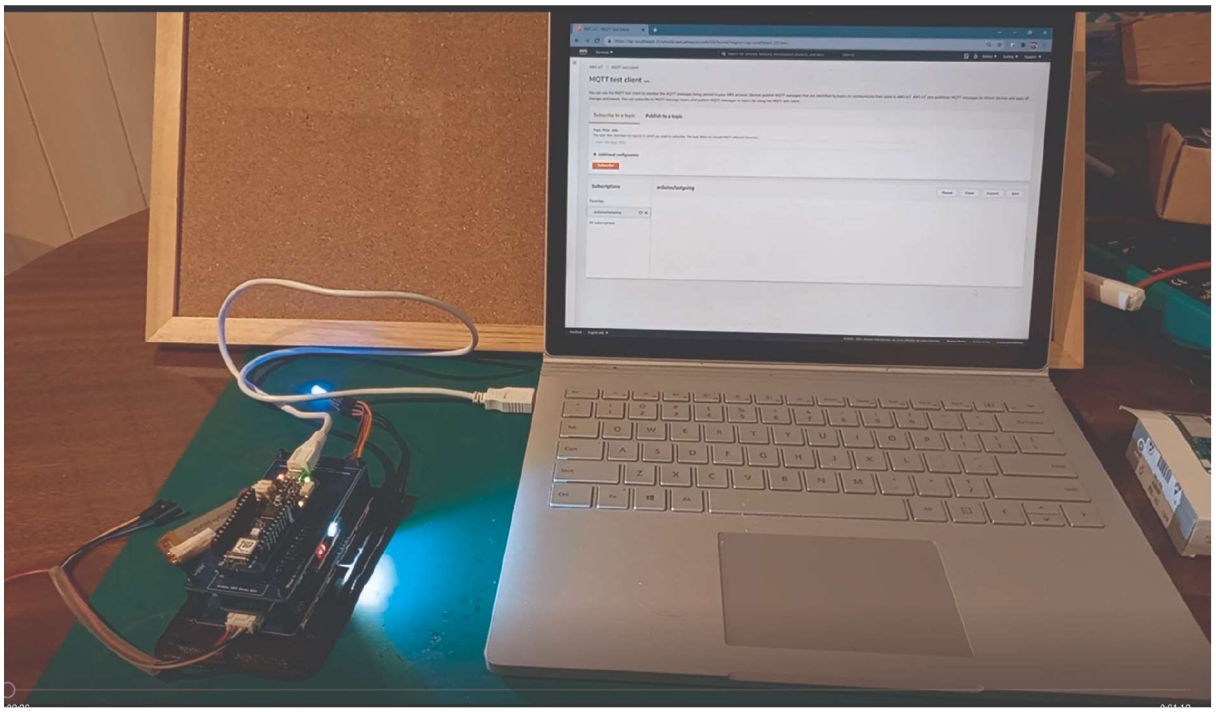
8.4 Project Presentation

To see a larger version of the entire series of Presentation Slides refer to **Appendix H.**

Project Presentation Slides (Figure 43)

A larger depiction of these slides is included in **Appendix H.**

8.5 Project Demonstration



Prototype V4 Being Demonstrated. (Figure 44)

8.6 Learning Outcomes

A listing of some Modern Tools Usage during this project is provided in **Appendix J.**
A listing of some of my Graduate Attributes is provided in **Appendix K.**
Some additional learning outcomes are listed in **Appendix L.**

8.7 Project Summary

The purpose of this project was to build prototype versions of an electronic device designed for use by vendors in the restaurants, cafés and hospitality industry to connect customers' loyalty cards and/or mobile apps with the digital infrastructure of the vendor. Use cases may extend to include many types of retail stores.

8.8 Recommendations and Conclusion

The focus of this project has been on developing device hardware and firmware while leaving the development of software applications outside of the project scope. The exceptions to this were the configuration of Cloud server infrastructure and security features to enable the device to exchange message traffic with an AWS IoT Core secure server. This successfully completes the work in developing multiple proof-of-concept prototype devices that demonstrate the potential of investing in building solutions with these kinds of technologies.

List of Figures

All photographs, diagrams and screenshots are by the author unless otherwise indicated.

List of Tables

All tables are by the author unless otherwise indicated.

List of Charts

All charts are by the author unless otherwise indicated.

References

Adafruit (03/02/2016). *Adafruit DS3231 Precision I2C RTC Breakout.* Retrieved from https://learn.adafruit.com/adafruit-ds3231-precision-rtc-breakout?view=all

Arduino.cc (n.d.). *analogWrite() - Arduino Reference.* Retrieved from https://www.arduino.cc/reference/en/language/functions/analog-io/analogwrite/

Arduino.cc (n.d.). *ArduinoBLE Library.* Retrieved from https://www.arduino.cc/en/Reference/ArduinoBLE

Arduino.cc (n.d.). *Arduino MKR WiFi 1010.* Retrieved from https://store.arduino.cc/usa/mkr-wifi-1010

Arduino.cc (n.d.). *MKR2UNO Adapter.* Retrieved from https://store.arduino.cc/usa/mkr2uno-adapter

Arduino.cc (n.d.). *Nano 33 IOT.* Retrieved from https://store.arduino.cc/usa/nano-33-iot

Arduino.cc (n.d.). *Nano 33 IOT Guide.* Retrieved from https://www.arduino.cc/en/Guide/NANO33IoT

Arduino.cc (n.d.). *NTPClient: Connect to a NTP server.* Retrieved from https://github.com/arduino-libraries/NTPClient

Arduino.cc (n.d.). *WiFiNINA Library.* Retrieved from https://www.arduino.cc/en/Reference/WiFiNINA

Arduino.cc (n.d.). *WiFi.getTime().* Retrieved from https://www.arduino.cc/en/Reference/WiFiNINAGetTime

Arduino.cc (24/08/2016). *MKR2UNO Schematic v3.0.* Retrieved from https://www.arduino.cc/en/uploads/Main/MKR2UNO-schematic.pdf

Arduino.cc (05/02/2018). *Getting started with the Arduino Nano 33 IoT.* Retrieved from https://www.arduino.cc/en/Guide/NANO33IoT

Arduino.cc (05/02/2018). *Adding more Serial Interfaces to SAMD microcontrollers (SERCOM).* Retrieved from https://www.arduino.cc/en/Tutorial/SamdSercom

Arduino.cc (24/12/2019). *RTC library.* Retrieved from https://www.arduino.cc/en/Reference/RTC

Arduino.cc (07/08/2020). *Arduino MKR WiFi 1010 Pinout.* Retrieved from https://content.arduino.cc/assets/Pinout-MKRwifi1010_latest.pdf

Arduino.cc (n.d.) *Arduino Time Library.* Retrieved from https://playground.arduino.cc/Code/Time/

Arduino_Genuino (16/01/2019). *Securely connecting an Arduino MKR WiFi 1010 to AWS IoT Core.* Retrieved from https://create.arduino.cc/projecthub/Arduino_Genuino/securely-connecting-an-arduino-mkr-wifi-1010-to-aws-iot-core-a9f365

AWS (n.d.). *AWS IoT.* Retrieved from https://ap-southeast-2.console.aws.amazon.com/iot/home

AWS (n.d.). *AWS IoT Core.* Retrieved from https://aws.amazon.com/iot-core/

AWS (n.d.). *AWS IoT Core - Developer Guide.* Retrieved from https://docs.aws.amazon.com/iot/latest/developerguide/iot-dg.pdf#iot-ddb-rule

China Young Sun LED Technology Co. Ltd (n.d.). *Red/Green/Blue triple color LED datasheet.* Retrieved from https://www.sparkfun.com/datasheets/Components/YSL-R596CR3G4B5C-C10.pdf

Digi-Key (n.d.). *LED series resistance calculator.* Retrieved from https://www.digikey.com/en/resources/conversion-calculators/conversion-calculator-led-series-resistor

DongGuan DYscan Technology Co. Ltd (n.d.). *2D Scan Engine DE2120 Specification.* Retrieved from https://nz.mouser.com/datasheet/2/813/DY_SCAN_Specification_DE2120_I___2_-1901308.pdf

Electronics Notes (n.d.). *Standard Resistor Values: E3, E6, E12, E24, E48 & E96.* Retrieved from https://www.electronics-notes.com/articles/electronic_components/resistors/standard-resistor-values-e-series-e3-e6-e12-e24-e48-e96.php

Electronics Tutorials (n.d.). *Pull-up Resistors.* Retrieved from
https://www.electronics-tutorials.ws/logic/pull-up-resistor.html

Fairchild Semiconductor (Oct 2005). *BSS138 N-Channel Logic Level Enhancement Mode FET.* Retrieved from
http://cdn.sparkfun.com/datasheets/BreakoutBoards/BSS138.pdf

Google (n.d.). *Google Public NTP.* Retrieved from
https://developers.google.com/time

Google (n.d.). *Making every (leap) second count with our new public NTP servers.* Retrieved from
https://cloud.google.com/blog/products/gcp/making-every-leap-second-count-with-our-new-public-ntp-servers

Guadalupi, Arturo (28/01/2019). *MKRWiFi1010v2.0 Schematic.* Retrieved from
https://content.arduino.cc/assets/MKRWiFi1010V2.0_sch.pdf

Gupta, Sourav (29/11/2018). *What is pull up and pull down resistor and where to use them.* Retrieved from
https://circuitdigest.com/tutorial/pull-up-and-pull-down-resistor

Hamblen, Jim (28/08/2020). *Using RGB LEDs.* Retrieved from
https://os.mbed.com/users/4180_1/notebook/rgb-leds/

Hirzel, Timothy (05/05/2018). *Pulse Width Modulation (PWM). Retrieved from*
https://www.arduino.cc/en/Tutorial/Foundations/PWM

Hobby Hour (n.d.). *LED series resistor calculator.* Retrieved from
https://www.hobby-hour.com/electronics/ledcalc.php

ITEAD (2021). *NX4024T032 – Nextion 3.2" Basic Series HMI Touch Display.* Retrieved from
https://itead.cc/product/nextion-nx4024t032-generic-3-2-hmi-tft-lcd-touch-display-module/

Itead.cc (n.d.). *Nextion NNX4024T032 3.2" HMI LCD TFT Screen Display Module.* Retrieved from
https://www.itead.cc/nextion-nx4024t032.html

Internet.nz (n.d.) .nz NTP Network. Retrieved from
https://ntp.net.nz/

ITP Physical Computing (). *Lab: Using a Real-Time Clock (RTC).* Retrieved from
https://itp.nyu.edu/physcomp/labs/lab-using-a-real-time-clock/

Julian, Stephen (May 2018). *DE6408 - Electronics Manufacturing 2 - Project Report.* Retrieved from author's Unitec academic archive.

Loladia, Rameez (20/05/2016). *Elliptic Curve Cryptography and Forward Secrecy Support in AWS.* Retrieved from
https://aws.amazon.com/blogs/iot/elliptic-curve-cryptography-and-forward-secrecy-support-in-aws-iot-3/

Monolithic Power Systems (MPS) (01/07/2015). *MPM3610 DCDC Converter Datasheet.* Retrieved from
https://www.monolithicpower.com/en/documentview/productdocument/index/version/2/document_type/Datasheet/lang/en/sku/MPM3610GQV/document_id/2090

Mouser (n.d.). *SparkFun SPX-16441.* Retrieved from
https://www.mouser.com/access/?pn=SPX-16441

Mouser (n.d.). *Adafruit DS3231 RTC.* Retrieved from
https://nz.mouser.com/datasheet/2/737/adafruit_ds3231_precision_rtc_breakout-1396544.pdf

Nextion (n.d.). *Nextion Basic Series Introduction.* Retrieved from
https://nextion.tech/basic-series-introduction/

Nextion (n.d.). *Nextion Editor.* Retrieved from
https://nextion.tech/nextion-editor/

Nextion (n.d.). *NX4024T032 Datasheet.* Retrieved from
https://nextion.tech/datasheets/nx4024t032/

ntp.org (n.d.) *NTP: The Network Time Protocol.* Retrieved from
https://www.ntp.org/

Ohms Law Calculator (). *E24 resistor sizes.* Retrieved from
https://ohmslawcalculator.com/e24-resistor-sizes

Online Barcode Generator (n.d.). *Online Barcode Generator.* Retrieved from
https://barcode.tec-it.com/en

Ostaquet (n.d.). *Arduino Nano 33 IoT - Ultimate Guide.* Retrieved from
https://github.com/ostaquet/Arduino-Nano-33-IoT-Ultimate-Guide

Savaşcı, Tamer (28/04/2017). *Programming Nextion LCD without SD card.* Retrieved from
https://create.arduino.cc/projecthub/tsavascii/programming-nextion-lcd-without-sd-card-6b18d7

Seitanis, Thanisis (02/02/2021). *EasyNextionLibrary.* Retrieved from https://github.com/Seithan/EasyNextionLibrary

Seitanis, Athanasios (n.d.). *Easy Nextion Library.* Retrieved from https://seithan.com/Easy-Nextion-Library/

Seitanis, Athanasios (n.d.). *Tutorial on how to write code with Nextion and Arduino.* Retrieved from https://seithan.com/projects/nextion-tutorial/

SparkFun Electronics (n.d.). *2D Barcode Scanner Breakout - SPX-16441.* Retrieved from https://www.sparkfun.com/products/16441

SparkFun Electronics (n.d.). *2D Barcode Scanner Schematic.* Retrieved from https://cdn.sparkfun.com/assets/3/8/9/6/0/2D_Barcode_Scanner.pdf

SparkFun Electronics (n.d.). *2D Barcode Scanner Settings Manual.* Retrieved from https://cdn.sparkfun.com/assets/b/5/0/e/e/DY_Scan_Setting_Manual-DE2120___19.4.6___.pdf

SparkFun Electronics (n.d.). *Bi-Directional Logic Level Converter Hookup Guide.* Retrieved from https://learn.sparkfun.com/tutorials/bi-directional-logic-level-converter-hookup-guide

SparkFun Electronics (n.d.). *Logic Level Converter - Bi-Directional.* Retrieved from https://www.sparkfun.com/products/12009

SparkFunX (n.d.). *2D_Barcode_Scanner.* Retrieved from https://github.com/sparkfunX/2D_Barcode_Scanner

Time Tools Limited (02/05/2018) *A guide to GPS NTP servers for network time synchronization.* Retrieved from https://www.ntp.org/

Time Tools Limited (06/12/2018) *An introduction to NTP.* Retrieved from https://timetoolsltd.com/ntp/what-is-ntp/

Time Tools Limited (06/12/2018) *What is GPS Time?* Retrieved from https://timetoolsltd.com/gps/what-is-gps-time/

Waveshare (n.d.). *2.8inch TFT Touch Shield.* Retrieved from https://www.waveshare.com/wiki/2.8inch_TFT_Touch_Shield

Waveshare (n.d.). *2.8inch Touch LCD Shield for Arduino.* Retrieved from https://www.waveshare.com/2.8inch-TFT-Touch-Shield.htm

Waveshare (n.d.). *Barcode Scanner Module, ID/2D Codes Reader.* Retrieved from https://www.waveshare.com/barcode-scanner-module.htm

Waveshare (10/10/2019). *Barcode Scanner Module User Manual v 1.2.* Retrieved from https://www.waveshare.com/w/upload/d/dd/Barcode_Scanner_Module_Setting_Manual_EN.pdf

Waveshare (n.d.). *Barcode Scanner Module, ID/2D Codes Reader.* Retrieved from https://www.waveshare.com/barcode-scanner-module.htm

Appendices

Appendix	Title
A	Project Definition - Proposal for Student Project (MG7101)
B	Firmware for QRScan V4 (as used in Prototypes)
C	Third Party Firmware Libraries (as used in Prototypes)
D	Circuit Schematics of PCB V1, PCB V2
E	3D Design Renderings of PCB V1, PCB V2 and PCB V3
F	Bill of Materials (BOM) for Prototypes V1, V3, V4
G	Gantt Charts - Semesters A and B
H	Project Presentation Slides
I	Project Poster
J	Modern Tools Usage by Stephen Julian in this project
K	Graduate Attributes
L	(1.) Cover Letter from Stephen Julian, followed by (2.) Curriculum Vitae (CV) of Stephen Julian.

Appendix A

Project Definition - Proposal for Student Project (MG7101)

PROPOSAL FOR STUDENT PROJECT (MG7101)

Project:	**Sign-in Kiosk for Retail Customer Loyalty Programme**	
Category:	**Electronics, Microcontrollers, IoT**	**Level: 7**
Description:	*Briefly describe the scope and importance of project* This project is to produce a prototype or series of prototypes of a device designed for use as part of a customer loyalty programme in a retail environment. The device will feature a microcontroller, a QR code reader module and BT/BLE and WiFi for connectivity with a Tablet and/or Cloud server. Other components may include PIR, LDR, RGB LED, piezoelectric Buzzer and Touchscreen LCD display. A custom PCB will be designed and manufactured and an enclosure will be designed and 3D printed should time permit.	
Supervisor:	Morgan Look	
Allocated to students:	Stephen Julian	
Hazards:	*List Lab or Workshop related Hazards for H&S purposes* No known hazards.	
Industry Relevance:	*This is important for external moderation by NZQA or ENZ* The current pandemic has resulted in widespread familiarity and experience with, and use of, QR codes in the community. This removes inhibiting technical barriers for end users to encourage the inclusion of QR code scanning technologies as a part of customer loyalty programmes in the restaurant, cafe, hospitality and retail industries. The developed solution may have industry relevance. IP to be developed under the open source MIT licence.	
Funding:	The student already has most of the required components. Remaining costs of materials: 1) Suitable QR code reader OEM module (US$50 to 90) +pp 2) Suitable touchscreen LCD display module (US$28 to 45) +pp Manufacturing costs: 1) Manufacturing of the PCB. 2) 3D printing of the enclosure.	
Preferred Lab Location:	TBC	
Student Achievements:	*List expected knowledge/skills students will gain from the project* Practical experience working with Micros, Sensors, Actuators, BT/BLE and WiFi connectivity, Cloud, Barcode/QR Code, SPI, I2C, HID, POS, PCB design, 3D CAD and 3D Printing.	

Appendix B

Firmware for QRScan V4 (as used in Prototype V4)

mg7101_prototype_v4 § | arduino_secrets.h

```
/*
 MG7101 QR Code Scanning Kiosk Prototype V4
 Sync time of DS321 RTC with Network Time Protocol (NTP) time server.
 Scan a 1D barcode, 2D barcode or QR code and upload date/time and scanned value to AWS IoT Core server.
 created 4 Sep 2021
 by Stephen Julian
 modified 11 Nov 2021
 by Stephen Julian
 Includes some code snippets adapted from the example sketches bundled with include libraries.
 These libraries are included below and described in the project Final Report document.
*/

int SenseFromPIRPin = 4;
int TriggerToQRScanPin = 20;
int StatusFromQRScanPin = 21;
int RxFromQRScanPin = 3; //mySerial1 (QR Code Scanner)
int TxToQRScanPin = 2; //mySerial1 (QR Code Scanner)
int RxFromLcdPin = 1; //mySerial2 (LCD Display)
int TxToLcdPin = 0; //mySerial2 (LCD Display)
int LEDredPin = 7;
int LEDbluePin = 6;
int LEDgreenPin = 5;
bool ledIsFlashing = false;
bool ledIsOn = false;
int val = 0;
bool result = false;
bool modeScan = false;
bool debugMode = false;
bool firstNTPSyncDone = false;

#include <Arduino.h>
#include <wiring_private.h>
#include "SparkFun_DE2120_Arduino_Library.h" //Get the library here: http://librarymanager/All#SparkFun_DE2120
#include <stdio.h> // used for basic concatenating of strings in C
#include <string.h> // used for basic concatenating of strings in C
#include <SPI.h>
#include <WiFiNINA.h>
#include <WiFiUdp.h>
#include <RTClib.h>
#include <ArduinoBearSSL.h>
#include <ArduinoECCX08.h>
#include <ArduinoMqttClient.h>
#include "arduino_secrets.h"

// UARTs with SERCOM configs compatible with Nano 33 IoT. See docs:
// https://github.com/arduino/ArduinoCore-samd/blob/master/cores/arduino/SERCOM.h
// https://github.com/arduino/ArduinoCore-samd/blob/master/variants/nano_33_iot/variant.h
// https://github.com/arduino/ArduinoCore-samd/blob/master/variants/nano_33_iot/variant.cpp
//Uart mySerial1(&sercom0, 5, 6, SERCOM_RX_PAD_1, UART_TX_PAD_0); // QR Code Reader serial interface.
//Uart mySerial2(&sercom1, RxFromLcdPin, TxToLcdPin, SERCOM_RX_PAD_1, UART_TX_PAD_2); // LCD Module serial interface.
// Spare serial interface. Create the new UART instance assigning it to pin 1, 0:
//Uart Serial1 (&sercom5, PIN_SERIAL1_RX, PIN_SERIAL1_TX, PAD_SERIAL1_RX, PAD_SERIAL1_TX);

// UARTs with SERCOM configs compatible with both MKR WiFi 1010 and MKR GSM 1400. See docs:
// https://github.com/arduino/ArduinoCore-samd/blob/master/cores/arduino/SERCOM.h
// https://github.com/arduino/ArduinoCore-samd/blob/master/variants/mkrwifi1010/variant.h
// https://github.com/arduino/ArduinoCore-samd/blob/master/variants/mkrwifi1010/variant.cpp
// TX pad is always either UART_TX_PAD_0 or UART_TX_PAD_2
Uart mySerial0 (&sercom0, 3, 2, SERCOM_RX_PAD_3, UART_TX_PAD_2); // RX=3, TX=2 // QR Code Reader
//Uart mySerial3 (&sercom3, 1, 0, SERCOM_RX_PAD_1, UART_TX_PAD_0); // RX=1, TX=0 // LCD Module
//Uart Serial1 (&sercom5, PIN_SERIAL1_RX, PIN_SERIAL1_TX, PAD_SERIAL1_RX, PAD_SERIAL1_TX); // RX=13, TX=14

DE2120 scanner;
#define BUFFER_LEN 40
char scanBuffer[BUFFER_LEN];

WiFiClient    wifiClient;            // Used for the TCP socket connection.
BearSSLClient sslClient(wifiClient); // Used for SSL/TLS connection, integrates with ECC508 crypto IC.
MqttClient    mqttClient(sslClient); // Used for messaging AWS IoT Core server.

RTC_DS3231 rtc;
int status = WL_IDLE_STATUS;
int keyIndex = 0; // your network key Index number (needed only for WEP. WEP is insecure, use WPA2 instead.)
unsigned int localPort = 2390;      // local port to listen for UDP packets
IPAddress timeServer(216, 239, 35, 4); // time2.google.com NTP server
const int NTP_PACKET_SIZE = 48; // NTP time stamp is in the first 48 bytes of the message
byte packetBuffer[ NTP_PACKET_SIZE]; //buffer to hold incoming and outgoing packets

// A UDP instance to let us send and receive packets over UDP
WiFiUDP Udp;

const int LEDFlashInterval = 1000; // 500 ms or half a second.
int millisAtLastLEDwrite = 0;

const int millisInterval = 30 * 1000; // 30 * 1000 = 30000 = 30 seconds
int millisAtLastSync = 0;

///////please enter your sensitive data in the Secret tab/arduino_secrets.h
const char ssid[] = SECRET_SSID;    // your network SSID (name)
const char pass[] = SECRET_PASS;    // your network password (use for WPA, or use as key for WEP)
const char broker[] = SECRET_BROKER;
const char* certificate = SECRET_CERTIFICATE;
const int deviceID = 1234;
int ledStatusCode = 0;
/*
```

```
 * Possible ledStatusCode values are:
0 = Waiting for the server = Red, Solid.
1 = Server error = Red, Flashing.
2 = Systems Nominal = Green, Solid.
3 = Data is in transit = Green, Flashing.
4 = Message Sent = Blue, Solid.
5 = New Message Received = Blue, Flashing.
6 = Reserved = Orange, Solid.
7 = Reserved = Orange, Flashing.
 *
 */

// START Attach interrupt handler(s) to SERCOM serial port(s).
// SERCOM IRQ for QR Code Scanner
void SERCOM0_Handler()
{
  mySerial0.IrqHandler();
}
/*
// SECCOM IRQ for [additional serial device]
void SERCOM5_Handler()
{
  Serial1.IrqHandler();
}

// SERCOM IRQ for LCD Module
void SERCOM3_Handler()
{
  mySerial3.IrqHandler();
}
*/
// END Attach interrupt handler(s) to SERCOM serial port(s).

bool syncRTCwithNTPserver() {
  bool result = sendNTPpacket(timeServer); // send an NTP packet to a time server
  // wait to see if a reply is available
  delay(1000);
  if (Udp.parsePacket()) {
    result = setLedStatusCode(5);
    Serial.println("packet received");
    // We've received a packet, read the data from it
    Udp.read(packetBuffer, NTP_PACKET_SIZE); // read the packet into the buffer
    //the timestamp starts at byte 40 of the received packet and is four bytes,
    // or two words, long. First, extract the two words:
    unsigned long highWord = word(packetBuffer[40], packetBuffer[41]);
    unsigned long lowWord = word(packetBuffer[42], packetBuffer[43]);
    // combine the four bytes (two words) into a long integer
    // this is NTP time (seconds since Jan 1 1900):
    unsigned long secsSince1900 = highWord << 16 | lowWord;
    Serial.print("Seconds since Jan 1 1900 = ");
    Serial.println(secsSince1900);
    // now convert NTP time into everyday time:
    Serial.print("Unix time = ");
    // Unix time starts on Jan 1 1970. In seconds, that's 2208988800:
    const unsigned long seventyYears = 2208988800UL;
    // subtract seventy years:
    unsigned long epoch = secsSince1900 - seventyYears;
    time_t t = epoch;
    rtc.adjust(DateTime(year(t), month(t), day(t), hour(t), minute(t), second(t)));
    firstNTPSyncDone = true;
    millisAtLastSync = millis();
    // print Unix time:
    Serial.println(epoch);
    Serial.print(year(t));
    Serial.print("/");
    Serial.print(month(t));
    Serial.print("/");
    Serial.println(day(t));
    // print the hour, minute and second:
    Serial.print("The UTC time is ");       // UTC is the time at Greenwich Meridian (GMT)
    Serial.print((epoch  % 86400L) / 3600); // print the hour (86400 equals secs per day)
    Serial.print(':');
    if (((epoch % 3600) / 60) < 10) {
      // In the first 10 minutes of each hour, we'll want a leading '0'
      Serial.print('0');
    }
    Serial.print((epoch  % 3600) / 60); // print the minute (3600 equals secs per minute)
    Serial.print(':');
    if ((epoch % 60) < 10) {
      // In the first 10 seconds of each minute, we'll want a leading '0'
      Serial.print('0');
    }
    Serial.println(epoch % 60); // print the second
    delay(3000);
    result = sendEpochMessageToAWS(epoch);
  }
  result = setLedStatusCode(2);
  return result;
}

bool sendEpochMessageToAWS(unsigned long myEpoch) {
  if (WiFi.status() != WL_CONNECTED) {
    connectWiFi();
  }
  delay(2000);
```

```
  if (!mqttClient.connected()) {
    // MQTT client is disconnected, connect
    connectMQTT();
  }
  time_t t = myEpoch;
  DateTime rightnow = rtc.now();
  // poll for new MQTT messages and send keep alives
  mqttClient.poll();
  Serial.println("Publishing message...");
  // send message, the Print interface can be used to set the message contents
  mqttClient.beginMessage("arduino/outgoing");
  mqttClient.print("{");

  mqttClient.print("\"utc_datetime\":\"");
  mqttClient.print(rightnow.year(), DEC);
  mqttClient.print('/');
  mqttClient.print(rightnow.month(), DEC);
  mqttClient.print('/');
  mqttClient.print(rightnow.day(), DEC);
  mqttClient.print(" ");
  mqttClient.print(rightnow.hour(), DEC);
  mqttClient.print(':');
  mqttClient.print(rightnow.minute(), DEC);
  mqttClient.print(':');
  mqttClient.print(rightnow.second(), DEC);

  mqttClient.print(",");
  mqttClient.print("\"deviceID\":");
  mqttClient.print(deviceID);
  mqttClient.print(",");
  mqttClient.print("\"event\":");
  mqttClient.print("\"NTP server synchronised\",");
  mqttClient.print("\"epoch\":");
  mqttClient.print(myEpoch);
  mqttClient.print("}");
  mqttClient.endMessage();
  Serial.print("NTP Server synchronised. ");
  Serial.println("Message published!");
  Serial.println(" ");
  return setLedStatusCode(4);
}

bool sendScanMessageToAWS(char* myCode) {
  if (WiFi.status() != WL_CONNECTED) {
    connectWiFi();
  }
  if (!mqttClient.connected()) {
    // MQTT client is disconnected, connect
    connectMQTT();
  }
  // poll for new MQTT messages and send keep alives
  mqttClient.poll();

  Serial.println("Publishing message...");
  // send message, the Print interface can be used to set the message contents
  mqttClient.beginMessage("arduino/outgoing");
  mqttClient.print("{");
  mqttClient.print("\"utc_datetime\":\"");
  DateTime rightnow = rtc.now();

  mqttClient.print(rightnow.year(), DEC);
  mqttClient.print('/');
  mqttClient.print(rightnow.month(), DEC);
  mqttClient.print('/');
  mqttClient.print(rightnow.day(), DEC);
  mqttClient.print(" ");
  mqttClient.print(rightnow.hour(), DEC);
  mqttClient.print(':');
  mqttClient.print(rightnow.minute(), DEC);
  mqttClient.print(':');
  mqttClient.print(rightnow.second(), DEC);

  mqttClient.print("\",");
  mqttClient.print("\"deviceID\":");
  mqttClient.print(deviceID);
  mqttClient.print(",");
  mqttClient.print("\"Event\":");
  mqttClient.print("\"Code scanned.\",");
  mqttClient.print("\"Code\":\"");
  int imax = strlen(myCode);
  for (int i=0; i<imax; i++) {
    mqttClient.print(myCode[i]);
  }
  //mqttClient.print(myCode);
  mqttClient.print("\"");
  mqttClient.print("}");
  mqttClient.endMessage();
  //Serial.println(myEpoch);
  Serial.print("utc_datetime: ");
  Serial.print(rightnow.year());
  Serial.print(':');
  Serial.print(rightnow.month());
  Serial.print(':');
  Serial.print(rightnow.day());
  Serial.print(" ");
```

```
  Serial.print(rightnow.hour());
  Serial.print(':');
  Serial.print(rightnow.minute());
  Serial.print(':');
  Serial.print(rightnow.second());

  Serial.print(" Event: ");
  Serial.print("Code scanned. ");
  Serial.print("Code:");
  imax = strlen(myCode);
  for (int i=0; i<imax; i++) {
    Serial.print(myCode[i]);
  }
  Serial.print(". Message published!");
  Serial.println(" ");
  return setLedStatusCode(4);
}

// send an NTP request to the time server at the given address
unsigned long sendNTPpacket(IPAddress& address) {

  // set all bytes in the buffer to 0
  memset(packetBuffer, 0, NTP_PACKET_SIZE);
  // Initialize values needed to form NTP request
  // (see URL above for details on the packets)

  packetBuffer[0] = 0b11100011;   // LI, Version, Mode
  packetBuffer[1] = 0;     // Stratum, or type of clock
  packetBuffer[2] = 6;     // Polling Interval
  packetBuffer[3] = 0xEC;  // Peer Clock Precision

  // 8 bytes of zero for Root Delay & Root Dispersion
  packetBuffer[12]  = 49;
  packetBuffer[13]  = 0x4E;
  packetBuffer[14]  = 49;
  packetBuffer[15]  = 52;

  // all NTP fields have been given values, now
  // you can send a packet requesting a timestamp:
  Udp.beginPacket(address, 123); //NTP requests are to port 123
  Udp.write(packetBuffer, NTP_PACKET_SIZE);
  Udp.endPacket();
  return setLedStatusCode(4);
}

void printWifiStatus() {
  // print the SSID of the network you're attached to:
  Serial.print("SSID: ");
  Serial.println(WiFi.SSID());

  // print your board's IP address:
  IPAddress ip = WiFi.localIP();
  Serial.print("IP Address: ");
  Serial.println(ip);

  // print the received signal strength:
  long rssi = WiFi.RSSI();
  Serial.print("signal strength (RSSI):");
  Serial.print(rssi);
  Serial.println(" dBm");
}

unsigned long getTime() {
  // get the current time from the WiFi module
  return WiFi.getTime();
}

void connectWiFi() {
  Serial.print("Attempting to connect to SSID: ");
  Serial.print(ssid);
  Serial.print(" ");
  while (WiFi.begin(ssid, pass) != WL_CONNECTED) {
    // failed, retry
    Serial.print(".");
    delay(5000);
  }
  Serial.println();
  Serial.println("You're connected to the network");
}

void connectMQTT() {
  Serial.print("Attempting to MQTT broker: ");
  Serial.print(broker);
  Serial.println(" ");
  while (!mqttClient.connect(broker, 8883)) {
    // failed, retry
    Serial.print(".");
    delay(1000);
  }
  Serial.println();
  Serial.println("You're connected to the MQTT broker");
  // subscribe to a topic
  mqttClient.subscribe("arduino/incoming");
}
```

```
void onMessageReceived(int messageSize) {
  // we received a message, print out the topic and contents
  Serial.print("Received a message with topic '");
  Serial.print(mqttClient.messageTopic());
  Serial.print("', length ");
  Serial.print(messageSize);
  Serial.println(" bytes:");
  // use the Stream interface to print the contents
  while (mqttClient.available()) {
    Serial.print((char)mqttClient.read());
  }
  Serial.println();
  Serial.println();
}

bool setLedStatusCode(int statusCode) {
  ledStatusCode = statusCode;
  return true;
}

bool makeLEDOff() {
  analogWrite(LEDredPin, 0);
  analogWrite(LEDgreenPin, 0);
  analogWrite(LEDbluePin, 0);
  ledIsOn = false;
  delay(50);
  return true;
}

bool makeLEDOn() {
  result = false;
  switch(ledStatusCode) {
    case 0:
        //statements
        ledIsFlashing = false;
        result = makeLEDRed();
        break;
    case 1:
        //statements
        ledIsFlashing = true;
        result = makeLEDRed();
        break;
    case 2:
        //statements
        ledIsFlashing = false;
        result = makeLEDGreen();
        break;
    case 3:
        //statements
        ledIsFlashing = true;
        result = makeLEDGreen();
        break;
    case 4:
        //statements
        ledIsFlashing = false;
        result = makeLEDBlue();
        break;
    case 5:
        //statements
        ledIsFlashing = true;
        result = makeLEDBlue();
        break;
    case 6:
        ledIsFlashing = false;
        result = makeLEDOrange();
        //statements
        break;
    case 7:
        //statements
        ledIsFlashing = true;
        result = makeLEDOrange();
        break;
    default:
        ledIsFlashing = false;
        result = makeLEDRed();
  }
  ledIsOn = true;
  return result;
}

bool makeLEDGreen() {
  analogWrite(LEDredPin, 0);
  analogWrite(LEDgreenPin, 255);
  analogWrite(LEDbluePin, 0);
  delay(50);
  return true;
}

bool makeLEDRed() {
  analogWrite(LEDredPin, 255);
  analogWrite(LEDgreenPin, 0);
  analogWrite(LEDbluePin, 0);
  delay(50);
  return true;
}
```

```
bool makeLEDBlue() {
  analogWrite(LEDredPin, 0);
  analogWrite(LEDgreenPin, 0);
  analogWrite(LEDbluePin, 255);
  delay(50);
  return true;
}

bool makeLEDOrange() {
  analogWrite(LEDredPin, 128);
  analogWrite(LEDgreenPin, 128);
  analogWrite(LEDbluePin, 0);
  delay(50);
  return true;
}

void setup() {
  pinPeripheral(2, PIO_SERCOM); // QR Code Reader
  pinPeripheral(3, PIO_SERCOM); // QR Code Reader
  delay(20);
  // Open serial communications and wait for port to open:
  Serial.begin(115200);
  delay(10);
  Serial.println("Starting setup...");
  pinMode(TriggerToQRScanPin, OUTPUT);
  digitalWrite(TriggerToQRScanPin, HIGH);
  delay(300);
  pinMode(LEDredPin, OUTPUT);
  pinMode(LEDbluePin, OUTPUT);
  pinMode(LEDgreenPin, OUTPUT);
  delay(1000);
  result = makeLEDOff();
  result = setLedStatusCode(0);
  if (! rtc.begin()) {
    Serial.println("Couldn't find RTC");
    Serial.flush();
    abort();
  }
  if (rtc.lostPower()) {
    Serial.println("RTC lost power, let's set the time!");
    // When time needs to be set on a new device, or after a power loss, the
    // following line sets the RTC to the date & time this sketch was compiled
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
    // This line sets the RTC with an explicit date & time, for example to set
    // January 21, 2014 at 3am you would call:
    // rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));
  }

  // When time needs to be re-set on a previously configured device, the
  // following line sets the RTC to the date & time this sketch was compiled
  // rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  // This line sets the RTC with an explicit date & time, for example to set
  // January 21, 2014 at 3am you would call:
  // rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));

  // check for the WiFi module:
  if (WiFi.status() == WL_NO_MODULE) {
    Serial.println("Communication with WiFi module failed!");
    // don't continue
    while (true);
  }
  String fv = WiFi.firmwareVersion();
  if (fv < WIFI_FIRMWARE_LATEST_VERSION) {
    Serial.println("Please upgrade the firmware");
  }
  delay(10);
  pinPeripheral(2, PIO_SERCOM); // QR Code Reader
  pinPeripheral(3, PIO_SERCOM); // QR Code Reader
  delay(20);
  pinMode(TriggerToQRScanPin, OUTPUT);
  digitalWrite(TriggerToQRScanPin, HIGH);
  delay(300);
  // initialize scanner serial communication at 9600 bits per second:
  mySerial0.begin(9600);
  if (scanner.begin(mySerial0) == false) {

    // MKR WiFi 1010 SERCOM0 pins:
    pinPeripheral(2, PIO_SERCOM); // QR Code Reader
    pinPeripheral(3, PIO_SERCOM); // QR Code Reader
    // MKR WiFi 1010 SERCOM3 pins:
    //pinPeripheral(0, PIO_SERCOM); // LCD Module. Assign TX function to pin 0.
    //pinPeripheral(1, PIO_SERCOM); // LCD Module. Assign RX function to pin 1.

    // Nano 33 IoT SERCOM0 pins:
    //pinPeripheral(5, PIO_SERCOM_ALT);
    //pinPeripheral(6, PIO_SERCOM_ALT);
    // Nano 33 IoT SERCOM1 pins:
    //pinPeripheral(13, PIO_SERCOM);
    //pinPeripheral(8, PIO_SERCOM);
    Serial.println("Scanner did not respond. Please check!");
  }
  digitalWrite(TriggerToQRScanPin, HIGH);
  delay(200);
  Serial.print(scanner.startScan());
```

```
  delay(1);
   Serial.print(scanner.changeBaudRate(9600));
  delay(1);
   Serial.print(scanner.enableContinuousRead(3));
  delay(1);
   Serial.println(scanner.lightOff());
  delay(1);
   Serial.print(scanner.reticleOn());
  delay(1);
   Serial.print(scanner.startScan());
  delay(1);
   Serial.println(scanner.startScan());
  delay(200);
  //digitalWrite(TriggerToQRScanPin, HIGH);
  result = makeLEDGreen();
  delay(300);
  connectWiFi();
  Serial.println("Connected to wifi");
  printWifiStatus();
  Serial.println("\nStarting connection to server...");
  Udp.begin(localPort);

  // Check for presence of ECCX08 crypto IC
  if (!ECCX08.begin()) {
    Serial.println("No ECCX08 present!");
    while (1);
  }

  // Set a callback to get the current time
  // used to validate the servers certificate
  ArduinoBearSSL.onGetTime(getTime);

  // Set the ECCX08 slot to use for the private key
  // and the accompanying public certificate for it
  sslClient.setEccSlot(0, certificate);

  // Optional, set the client id used for MQTT,
  // each device that is connected to the broker
  // must have a unique client id. The MQTTClient will generate
  // a client id for you based on the millis() value if not set
  //
  // mqttClient.setId("clientId");

  // Set the message callback, this function is
  // called when the MQTTClient receives a message
  mqttClient.onMessage(onMessageReceived);
  result = setLedStatusCode(2);
  Serial.println("Setup finished.");
  delay(3000);
}

void loop() {
  int millisSinceLastLEDwrite = millis() - millisAtLastLEDwrite;
  if(!firstNTPSyncDone) {
    result = syncRTCwithNTPserver(); // trigger RTC time sync with the NTP server
    delay(250);
  } else if(millisSinceLastLEDwrite > LEDFlashInterval) { // check if due for an RTC time sync with NTP server
    if(ledIsOn) {
      result = makeLEDOff();
    } else {
      result = makeLEDOn();
    }
    if(millisSinceLastLEDwrite > (LEDFlashInterval*2)) {
      millisAtLastLEDwrite = millis(); // store marker for when we last completed an LED ON/OFF Flash cycle.
    }
    delay(250);
  } else {
    int millisSinceLastSync = millis() - millisAtLastSync; // get time since last RTC time sync with NTP server
    if(millisSinceLastSync > millisInterval) { // check if due for another RTC time sync with the NTP server
      result = syncRTCwithNTPserver(); // trigger RTC time sync with the NTP server
      millisAtLastSync = millis(); // store marker for when we last synced the RTC time with the NTP server.
    }
    delay(250);
  }
  digitalWrite(TriggerToQRScanPin, HIGH);
  delay(200);
  Serial.println(scanner.startScan());
  delay(1);
  Serial.println(scanner.changeBaudRate(9600));
  delay(1);
  Serial.println(scanner.enableContinuousRead(3));
  delay(1);
  Serial.println(scanner.lightOff());
  delay(1);
  Serial.println(scanner.reticleOn());
  delay(1);
  Serial.println(scanner.startScan());
  delay(1);
  Serial.println(scanner.startScan());

  digitalWrite(TriggerToQRScanPin, HIGH);
  delay(300);
  //Serial.println("start while loop");
  int maxCount = (int)millis() + 20000;
  while ((int)millis() < maxCount)
```

```
  Serial.println("\nStarting connection to server...");
  Udp.begin(localPort);

  // Check for presence of ECCX08 crypto IC
  if (!ECCX08.begin()) {
    Serial.println("No ECCX08 present!");
    while (1);
  }

  // Set a callback to get the current time
  // used to validate the servers certificate
  ArduinoBearSSL.onGetTime(getTime);

  // Set the ECCX08 slot to use for the private key
  // and the accompanying public certificate for it
  sslClient.setEccSlot(0, certificate);

  // Optional, set the client id used for MQTT,
  // each device that is connected to the broker
  // must have a unique client id. The MQTTClient will generate
  // a client id for you based on the millis() value if not set
  //
  // mqttClient.setId("clientId");

  // Set the message callback, this function is
  // called when the MQTTClient receives a message
  mqttClient.onMessage(onMessageReceived);
  result = setLedStatusCode(2);
  Serial.println("Setup finished.");
  delay(3000);
}

void loop() {
  int millisSinceLastLEDwrite = millis() - millisAtLastLEDwrite;
  if(!firstNTPSyncDone) {
    result = syncRTCwithNTPserver(); // trigger RTC time sync with the NTP server
    delay(250);
  } else if(millisSinceLastLEDwrite > LEDFlashInterval) { // check if due for an RTC time sync with NTP server
    if(ledIsOn) {
      result = makeLEDOff();
    } else {
      result = makeLEDOn();
    }
    if(millisSinceLastLEDwrite > (LEDFlashInterval*2)) {
      millisAtLastLEDwrite = millis(); // store marker for when we last completed an LED ON/OFF Flash cycle.
    }
    delay(250);
  } else {
    int millisSinceLastSync = millis() - millisAtLastSync; // get time since last RTC time sync with NTP server
    if(millisSinceLastSync > millisInterval) { // check if due for another RTC time sync with the NTP server
      result = syncRTCwithNTPserver(); // trigger RTC time sync with the NTP server
      millisAtLastSync = millis(); // store marker for when we last synced the RTC time with the NTP server.
    }
    delay(250);
  }
  digitalWrite(TriggerToQRScanPin, HIGH);
  delay(200);
  Serial.println(scanner.startScan());
  delay(1);
  Serial.println(scanner.changeBaudRate(9600));
  delay(1);
  Serial.println(scanner.enableContinuousRead(3));
  delay(1);
  Serial.println(scanner.lightOff());
  delay(1);
  Serial.println(scanner.reticleOn());
  delay(1);
  Serial.println(scanner.startScan());
  delay(1);
  Serial.println(scanner.startScan());

  digitalWrite(TriggerToQRScanPin, HIGH);
  delay(300);
  //Serial.println("start while loop");
  int maxCount = (int)millis() + 20000;
  while ((int)millis() < maxCount)
  {
    if (scanner.readBarcode(scanBuffer, BUFFER_LEN) > 0)
    {
      Serial.print("Code scanned: ");
      int imax = strlen(scanBuffer);
      for (int i=0; i<imax; i++) {
        Serial.print(scanBuffer[i]);
      }
      Serial.println();
      result = sendScanMessageToAWS(scanBuffer);
      break;
    }
    delay(200);
  }
  scanner.stopScan();
  digitalWrite(TriggerToQRScanPin, LOW);
  delay(1500); // delay in between reads for stability
}
```

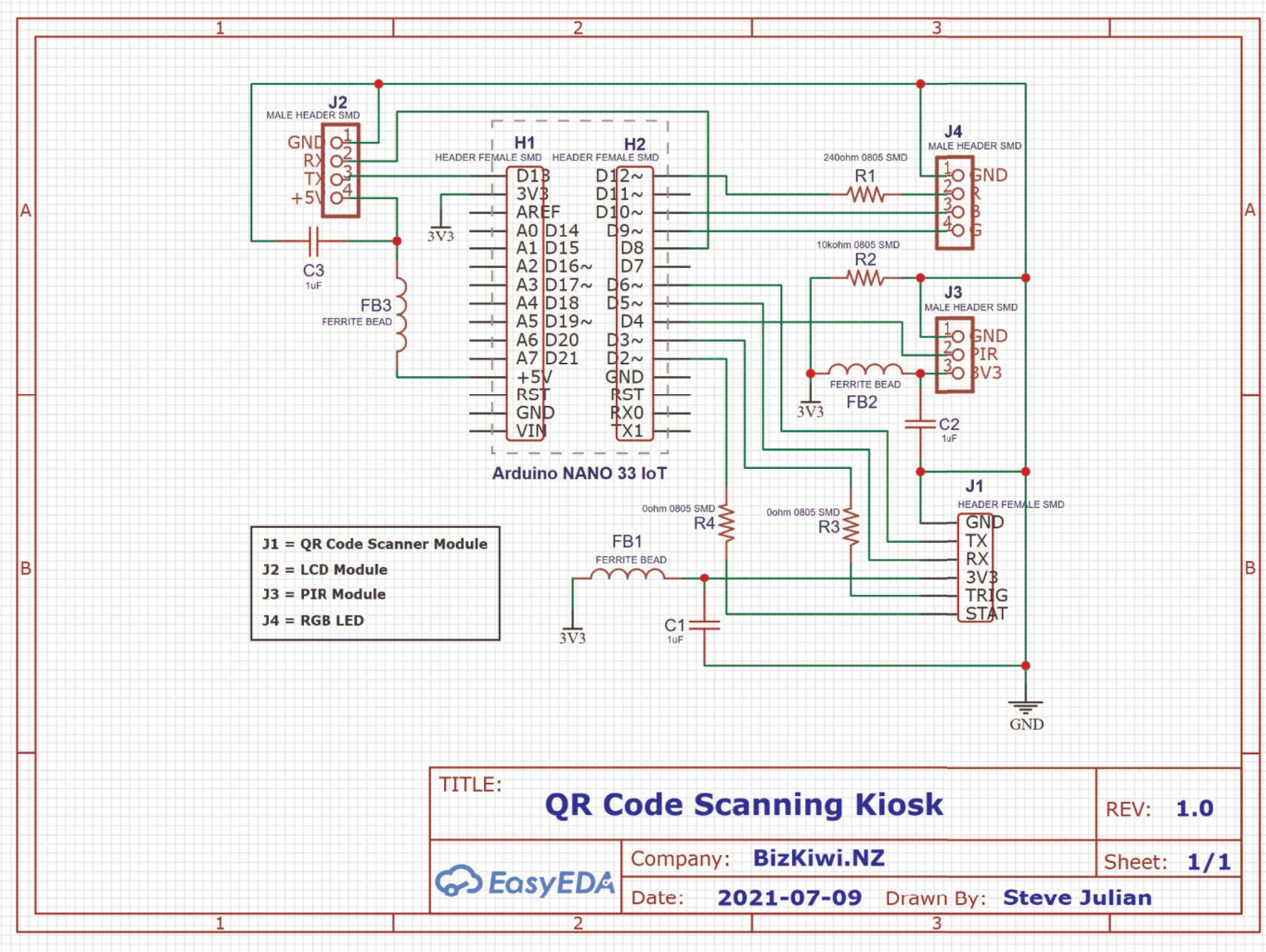
Appendix C

Third Party Firmware Libraries (as used in Prototypes)

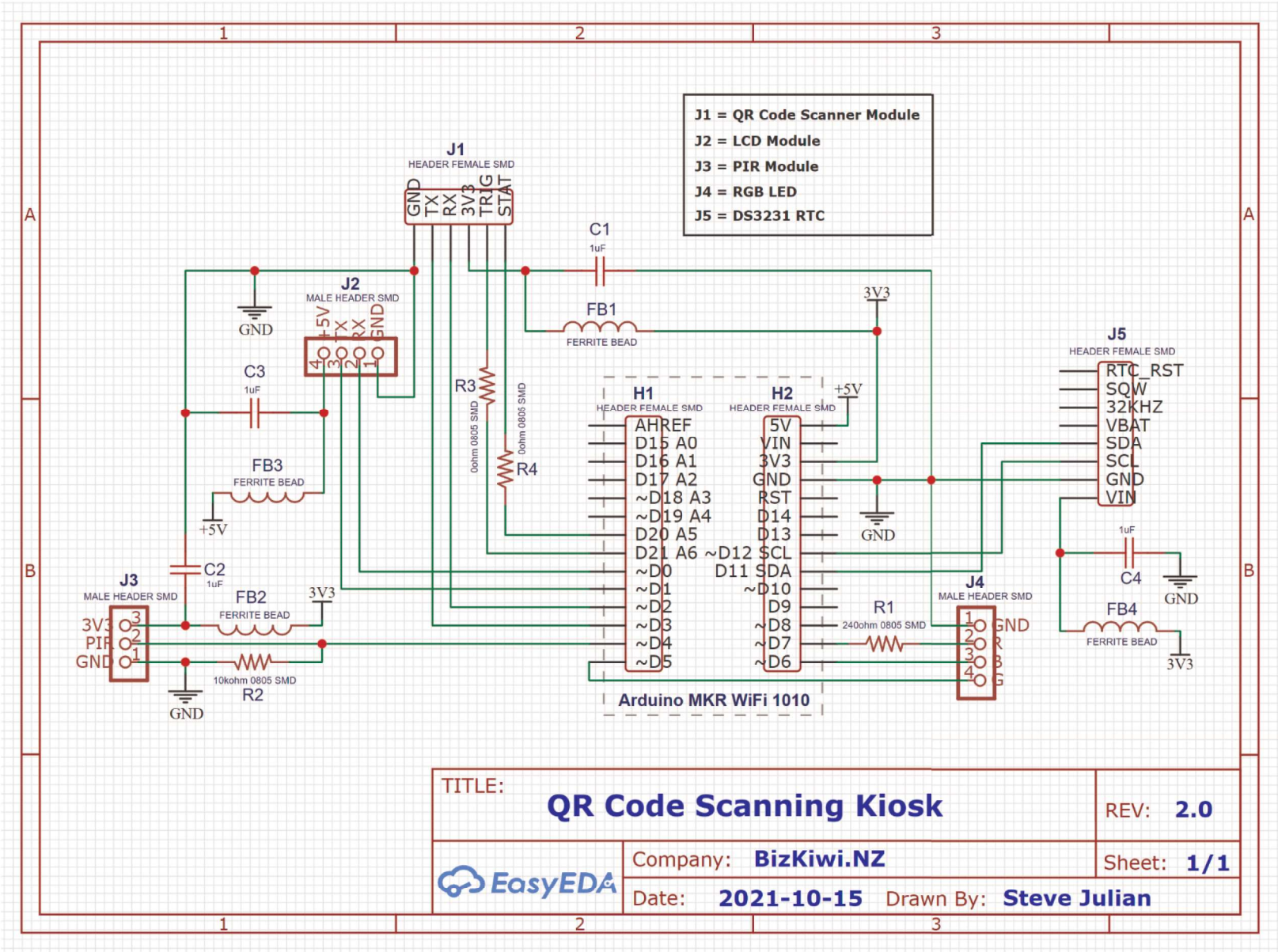
Code Library Header	Developer User	Function	GitHub Repository and Documentation	Prototypes						Licence
				V1	V2	V3	V4	V5	?	
<SparkFun_DE2I20_Arduino_Library.h>		Barcode and QR code scanning.	https://github.com/sparkfun/SparkFun_DE2I20_Arduino_Library https://github.com/sparkfunX/2D_Barcode_Scanner	✅	✅	✅	✅	✅	✅	GPL/MIT
<wiring_private.h>		For configuring SERCOM for multiple hardware serial ports.		✅	✅	✅	✅	✅	✅	NA
<EasyNextionLibrary.h>		Serial communication with Nextion LCD Module.	https://github.com/Seithan/EasyNextionLibrary	❌	✅	✅	✅	✅	✅	MIT
<trigger.h>		Installed with <EasyNextionLibrary.h>		❌	✅	✅	✅	✅	✅	MIT
<ArduinoBLE.h>		Bluetooth/BLE communications. May not be needed when restricted to use of USB and/or WiFi connectivity.	https://github.com/arduino-libraries/ArduinoBLE	❌	❌	❌	❌	❌	✅	NA
<WiFiNINA.h>		WiFi comms for Nano 33 IoT board.	https://github.com/arduino-libraries/WiFiNINA	❌	❌	✅	✅	✅	✅	NA
<SPI.h>		Communicates with SPI devices, with the Arduino as the master device such as WiFi radio.		❌	❌	❌	❌	❌	✅	NA
<WebUSB_Arduino.h>	reillyeon	Bidirectional comms between Arduino and website loaded in the Chrome web browser on a USB-connected Device.	https://github.com/webusb/arduino	❌	❌	❌	❌	❌	✅	NA
<RTClib.h> by Adafruit	Ubidefeo dherrada	Library for the DS3231 RTC. The Adafruit library is for the Adafruit DS3231 RTC module.	https://github.com/adafruit/RTClib https://github.com/NorthernWidget/DS3231	❌	❌	✅	✅	✅	✅	MIT
<NTPClient.h>	aentinger	Network Time Protocol (NTP) support for the RTC to be calibrated from an internet-connected atomic clock.	https://github.com/arduino-libraries/NTPClient	❌	❌	✅	✅	✅	✅	NA
<Time.h>	PaulStoffregen	Time and date calculations made simple.	https://github.com/PaulStoffregen/Time https://playground.arduino.cc/Code/Time/	❌	❌	✅	✅	✅	✅	NA
<Timezone_Generic.h>	Khoi Hoang	Timezone & Daylight Savings Time (DST) corrections.	https://github.com/khoih-prog/Timezone_Generic	❌	❌	❌	❌	❌	✅	GPL 3.0
<ArduinoBearSSL.h>	facchinm	TLS and SSL cryptography library for use with MQTT.	https://github.com/arduino-libraries/ArduinoBearSSL	❌	❌	❌	✅	✅	✅	MIT
<ArduinoMqttClient.h>	aentinger	Messaging protocol used with AWS IoT Core.	https://github.com/arduino-libraries/ArduinoMqttClient	❌	❌	❌	✅	✅	✅	LGPL-2.1
<ArduinioECCX08.h>	aentinger	MKR hardware storage of locked cryptographic keys.	https://github.com/arduino-libraries/ArduinoECCX08	❌	❌	❌	✅	✅	✅	GNU LGPL

Appendix D

Circuit Schematics of PCB V1 (Prototype V2)

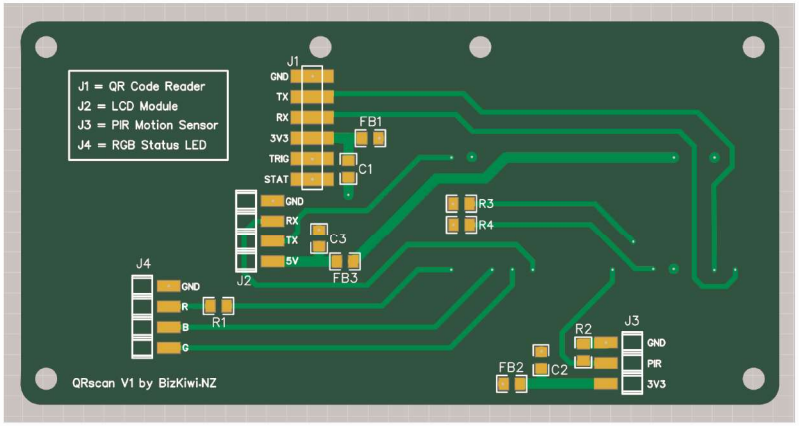


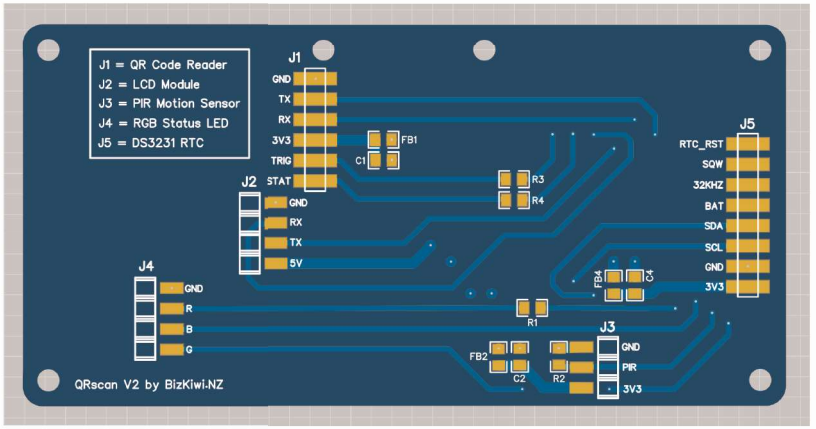
Circuit Schematics of PCB V2 (Prototype V3 and V4)

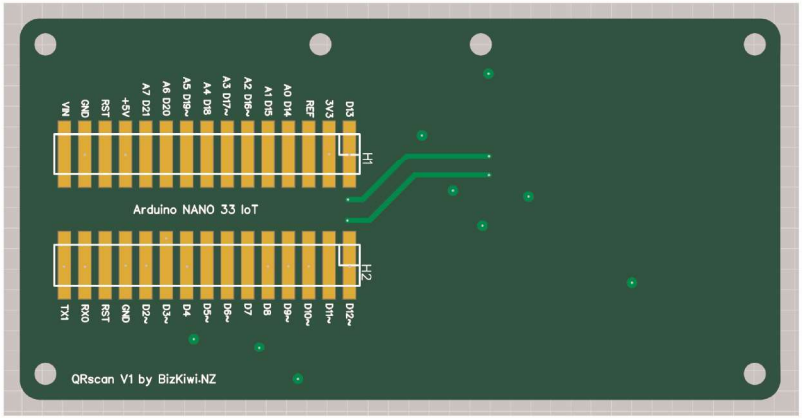


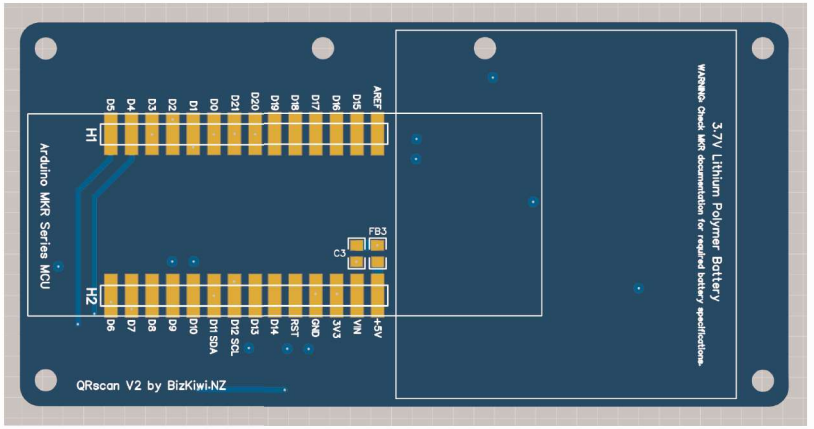
Appendix E

3D Design Renderings of PCB V1 (Prototype V2) and PCB V2 (Prototype V3 and V4)







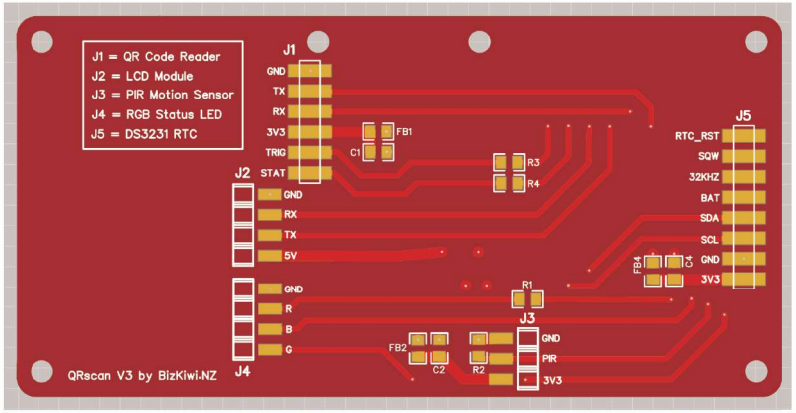


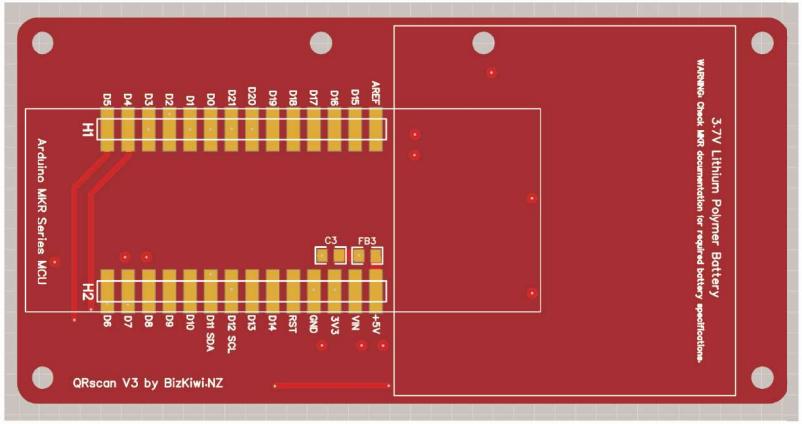
ABOVE LEFT: Top and Bottom Sides of PCB V1.

ABOVE RIGHT: Top and Bottom Sides of PCB V2.

Shown to approximate scale. Actual PCB Dimensions: 95 x 47.5 mm.

3D Design Renderings of PCB V3 (Prototype V5)





ABOVE: Top and Bottom Sides of PCB V3.

Shown to approximate scale. Actual PCB Dimensions: 95 x 47.5 mm.

Appendix F

Bill of Materials (BOM) for Prototype V1

Manufacturer	Supplier	Items	QTY	NZD/Unit	Sub-Total
Arduino.cc	Mouser	Arduino Nano 33 IoT 3.3V with TH Pin Headers	1	34.85	34.85
SparkFun	Mouser	DE2120 Barcode/QR Code Scanner Breakout Module, TH Pin Header.	1	88.20	88.20
NA	In House	AM312 PIR TH Module.	1	0.00	0.00
NA	In House	5mm TH Common Cathode RGB LED.	1	0.00	0.00
NA	In House	Resistors 1/8W TH	1	0.00	0.00
NA	In House	Breadboard	1	0.00	0.00
NA	In House	Wires	1	0.00	0.00
NA	In House	USB cable (male USB Type A to male Micro-USB)	1	0.00	0.00
				TOTAL (ex GST & Shipping) $	123.05

Prototype V1 - Bill of Materials (BOM)

Bill of Materials (BOM) for Prototype V3 and V4

Designator	Manufacturer	Manufacturer #	Supplier	Supplier #	Items	QTY	NZD/Unit	Sub-Total	Datasheet	Product
	Arduino.cc		rs-online.com		Arduino MKR WiFi 1010 3.3V with TH Pin Headers	1	53.00	53.00		
	SparkFun		Mouser		DE2120 Barcode/QR Code Scanner Breakout Module, TH Pin Header.	1	88.20	88.20		
	Nextion		Mouser		Nextion 3.2” LCD Module with connection wires.	1	68.50	68.50		
	NA		In House		AM312 PIR TH Module.	1	0.00	0.00		
	Adafruit		Mouser		Adafruit DS3231 Real Time Clock (RTC) Module	1	21.90	21.90		
	NA		Nice Gear		3.7V 1200mAh Lithium Polymer (LiPo) Rechargeable Battery	1	14.99	14.00		
	NA		Nice Gear		3V LiPo Coin Battery for RTC	1	2.50	2.50		
			In House		5mm TH Common Cathode RGB LED.	1	0.00	0.00		
			In House		Resistors 1/8W 0805 SMD		0.00	0.00		
			In House		Decoupling Capacitors, Ceramic 0805 SMD 0.1uF +-10% Rated 10V	4	0.00	0.00	Link	Link
			In House		Ferrite Bead, Power pin HF noise filters 0805 SMD 47ohm @ 100MHz	4	0.00	0.00	Link	Link
			In House		Female SMD Pin Header, 2.54mm pitch 1x3 pin (for AM312 PIR module)	1	0.00	0.00		
			In House		Female SMD Pin Header, 2.54mm pitch 1x6 pin (for DE2120 module)	1	0.00	0.00		
			In House		Female SMD Pin Header, 2.54mm pitch 2x15 pin (for MKR series board)	2	0.00	0.00		
			In House		Female SMD Pin Header, 2.54mm pitch 1x4 pin (for RGB LED)	1	0.00	0.00		
			In House		Female SMD Pin Header, 2.54mm pitch 1x8 pin (for DS3231 RTC)	1	0.00	0.00		
			In House		Nylon Spacers, Black 5mm diameter		0.00	0.00		
			In House		USB cable (male USB Type A to male Micro-USB)	1	0.00	0.00		
							TOTAL (ex GST & Shipping) $	**248.10**		
	Prototype V4 - Bill of Materials (BOM)									

Appendix G

Gantt Chart Actual Timeline - Semester A (Research Phase)

QR Code Scanning Kiosk

PROJECT TITLE	QR Code Scanning Kiosk
PROJECT MANAGER	Stephen Julian
COMPANY NAME	BizKiwi.NZ
DATE	27/07/2021

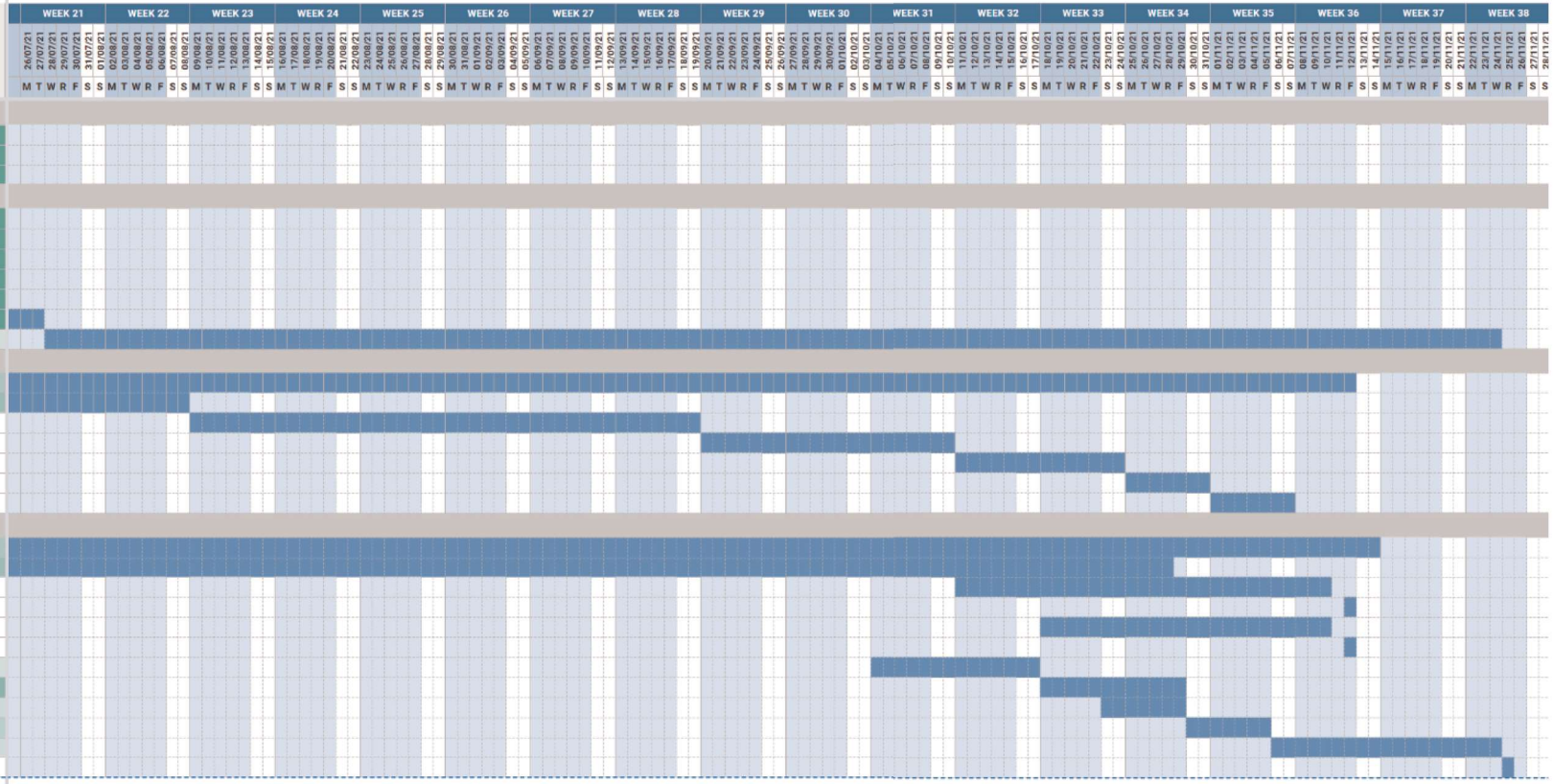
1	**Project Conception and Initiation**					
1.1	Project Charter	Steve J	12/03/18	15/03/18	3	100%
1.2	Background Research	Steve J	15/03/21	18/04/21	33	100%
1.3	Project Initiation	Steve J	15/03/21	21/03/21	6	100%
2	**Project Definition and Planning**					
2.1	Scope and Goal Setting	Steve J	17/03/21	31/03/21	14	100%
2.2	Prototypes Methodology	Steve J	22/03/21	04/04/21	12	100%
2.3	Prototype V1 Plan and Budget	Steve J	29/03/21	04/07/21	95	100%
2.4	Prototype V2 Plan and Budget	Steve J	05/04/21	04/07/21	89	100%
2.5	Prototype V3 Plan and Budget	Steve J	07/04/21	04/07/21	87	100%
2.6	Interim Report Writing	Steve J	17/03/21	08/07/21	111	100%
2.7	Final Report Writing	Steve J	14/06/21	14/11/21	150	25%
3	**Project Development**					
3.1	Status Tracking and Chart Updates	Steve J	15/03/21	12/11/21	237	50%
3.2	Prototype V1 Development	Steve J	19/07/21	08/08/21	19	60%
3.3	Prototype V2 Development	Steve J	09/08/21	19/09/21	40	0%
3.4	Prototype V3 Development	Steve J	20/09/21	31/10/21	41	0%
3.5	Prototype V4 Development	Steve J	20/09/21	31/10/21	41	0%
3.6	Prototype V5 Development	Steve J	20/09/21	31/10/21	41	0%
3.7	Source Code Packaging	Steve J	01/11/21	07/11/21	6	0%
4	**Project Monitoring and Presenting**					
4.1	Project Objectives	Steve J	17/03/21	14/11/21	237	50%
4.2	Resource and Cost Tracking	Steve J	29/03/21	28/10/21	209	60%
4.3	Project Powerpoint Preparation	Steve J	11/10/21	21/10/21	10	0%
4.4	Project Oral Presentation	Steve J	22/10/21	23/10/21	1	0%
4.5	Project Poster Design and Production	Steve J	18/10/21	04/11/21	16	0%
4.6	Project Practical Demonstration	Steve J	05/11/21	06/11/21	1	0%
4.7	Graduate Attributes Report Writing	Steve J			0	25%
4.8	CV Preparation	Steve J			0	70%
4.9	Cover Letter Writing	Steve J			0	30%
4.10	Project Performance	Steve J			0	40%
4.11	Project Deliverables Preparation	Steve J			0	30%
4.12	Submission of Completed Deliverables	Steve J	25/11/21	26/11/21	1	0%

Gantt Chart Projected Timeline - Semester B (Practical Phase)

QR Code Scanning Kiosk

PROJECT TITLE	QR Code Scanning Kiosk
PROJECT MANAGER	Stephen Julian
COMPANY NAME	BizKiwi.NZ
DATE	27/07/2021

1	**Project Conception and Initiation**					
1.1	Project Charter	Steve J	12/03/18	15/03/18	3	100%
1.2	Background Research	Steve J	15/03/21	18/04/21	33	100%
1.3	Project Initiation	Steve J	15/03/21	21/03/21	6	100%
2	**Project Definition and Planning**					
2.1	Scope and Goal Setting	Steve J	17/03/21	31/03/21	14	100%
2.2	Prototypes Methodology	Steve J	22/03/21	04/04/21	12	100%
2.3	Prototype V1 Plan and Budget	Steve J	29/03/21	04/07/21	95	100%
2.4	Prototype V2 Plan and Budget	Steve J	05/04/21	04/07/21	89	100%
2.5	Prototype V3 Plan and Budget	Steve J	07/04/21	04/07/21	87	100%
2.6	Interim Report Writing	Steve J	17/03/21	08/07/21	111	100%
2.7	Final Report Writing	Steve J	14/06/21	14/11/21	150	25%
3	**Project Development**					
3.1	Status Tracking and Chart Updates	Steve J	15/03/21	12/11/21	237	50%
3.2	Prototype V1 Development	Steve J	19/07/21	08/08/21	19	60%
3.3	Prototype V2 Development	Steve J	09/08/21	19/09/21	40	0%
3.4	Prototype V3 Development	Steve J	20/09/21	31/10/21	41	0%
3.5	Prototype V4 Development	Steve J	20/09/21	31/10/21	41	0%
3.6	Prototype V5 Development	Steve J	20/09/21	31/10/21	41	0%
3.7	Source Code Packaging	Steve J	01/11/21	07/11/21	6	0%
4	**Project Monitoring and Presenting**					
4.1	Project Objectives	Steve J	17/03/21	14/11/21	237	50%
4.2	Resource and Cost Tracking	Steve J	29/03/21	28/10/21	209	60%
4.3	Project Powerpoint Preparation	Steve J	11/10/21	21/10/21	10	0%
4.4	Project Oral Presentation	Steve J	22/10/21	23/10/21	1	0%
4.5	Project Poster Design and Production	Steve J	18/10/21	04/11/21	16	0%
4.6	Project Practical Demonstration	Steve J	05/11/21	06/11/21	1	0%
4.7	Graduate Attributes Report Writing	Steve J			0	25%
4.8	CV Preparation	Steve J			0	70%
4.9	Cover Letter Writing	Steve J			0	30%
4.10	Project Performance	Steve J			0	40%
4.11	Project Deliverables Preparation	Steve J			0	30%
4.12	Submission of Completed Deliverables	Steve J	25/11/21	26/11/21	1	0%

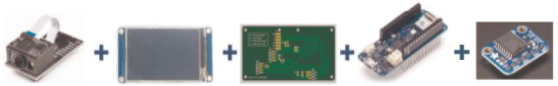


Appendix H

Project Presentation Slides: 1 to 4 of 16.

QR Code Scanning Kiosk

Development of a QR Code Scanning Kiosk
for use in a Retail Customer Loyalty Programme.



Author: Stephen Julian (Student ID: 1041664)
Bachelor of Engineering Technology (Electrical)
Department of Electrotechnology
Unitec Institute of Technology

1

Introduction

- Development of a Scanning Kiosk that:
- scans barcodes and QR codes,
- in a customer loyalty programme,
- lets shoppers collect points,
- trading points for rewards from
- restaurants, cafés, hospitality, retail

2

Background

Customer Loyalty Programme

Scan Code with Phone App to Collect Points

Ordinary Receipt

QR Code Receipt

Collect Food Rewards!

3

Methodology

Research & Development Strategy

A series of rapid prototypes,
each with features and functions
that improve on the previous version.

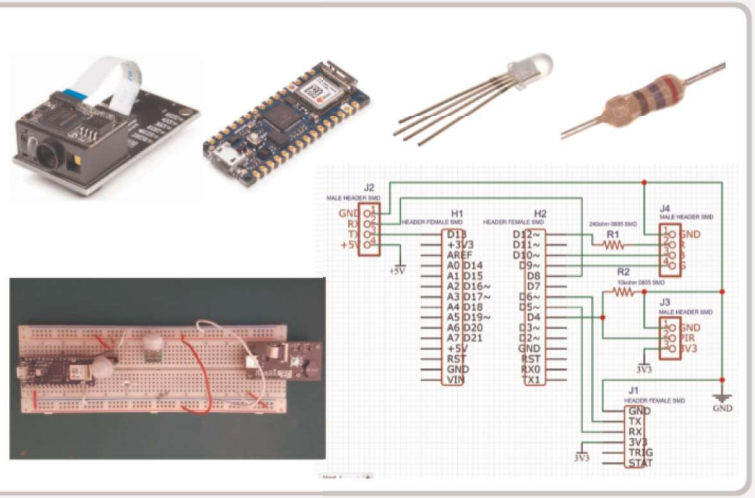
4

Project Presentation Slides: 5 to 8 of 16.

Prototype V1 - Breadboard Circuit

- Arduino Nano IoT 33 MCU
- QR Code Scanner Module
- PIR Motion Sensor
- RGB LED Status Indicator
- Miscellaneous Components
- Functional Testing, Proof of Concept

5

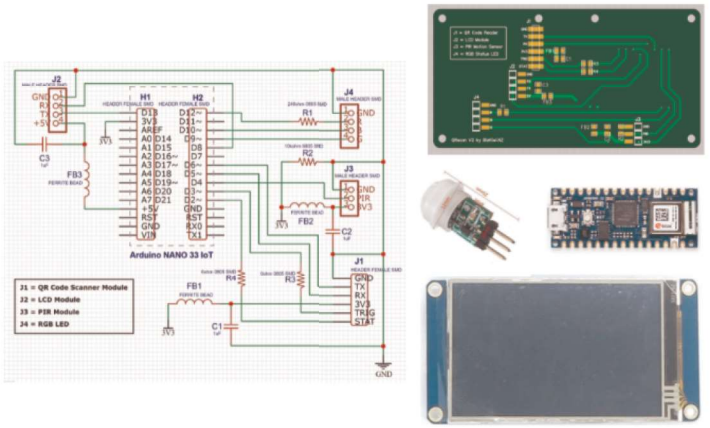


6

Prototype V2 - Custom PCB V1

- Arduino Nano IoT 33 MCU
- Add Custom PCB for Modules
- Add LCD Display Module
- Add Decoupling Capacitors
- Add Ferrite Beads as Filters

7



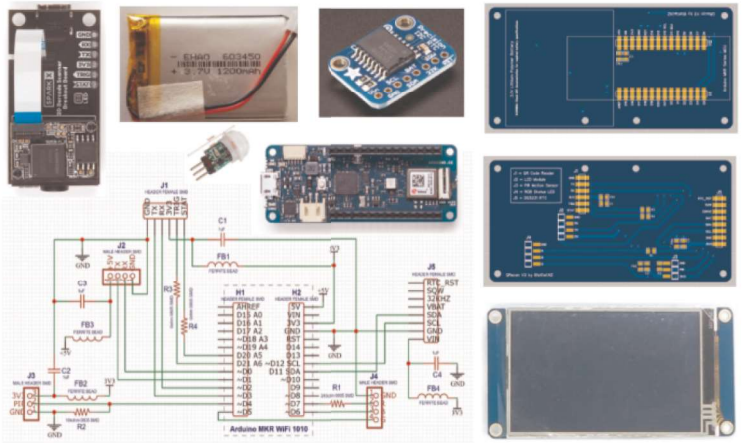
8

Project Presentation Slides: 9 to 12 of 16.

Prototype V3 - Custom PCB V2

- Swap MCU for MKR WiFi 1010
- Correctly Configure SERCOM
- Add LiPo Battery
- DS3231 Real-time Clock (RTC)
- Network Time Protocol (NTP)

9



10

Prototype V4 - Secure Cloud

- Add CSR Requests, x.509, CA Cert's
- Store Keys in Locked Crypto Chips
- Add ArduinoBearSSL
- Add MQTT
- AWS IoT Core:
  Things, Topics, Policies, Certificates

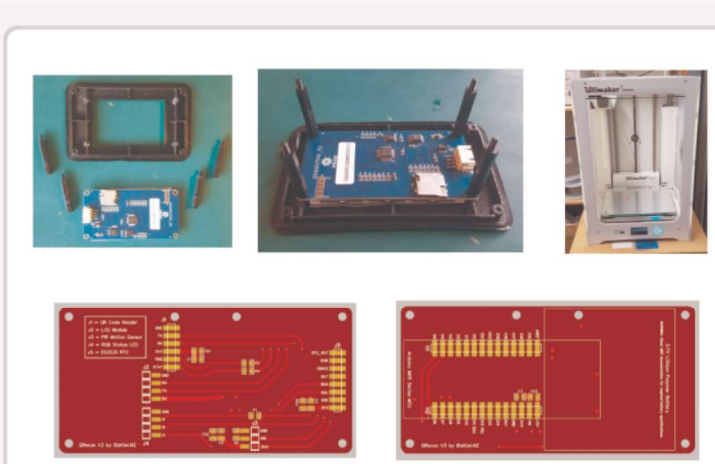
11

Prototype V5 - Final Touches

- Build a GUI for LCD Module
- Calibrate the Touchscreen
- Design & 3D Print Enclosure
- Assemble Unit in Enclosure
- PCB V3 with Design Improvements

12

Project Presentation Slides: 13 to 16 of 16.



13

Conclusions

Version 1 = Completed.

▼

Version 2 = Completed.

▼

Version 3 = Completed.

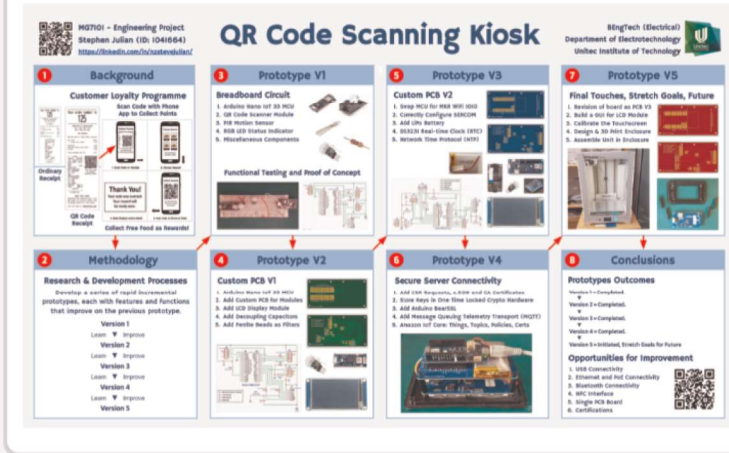
▼

Version 4 = Completed.

▼

Version 5 = Initiated with Stretch Goals for the Future

14



15

Thank you for your time.

Questions?

16

Appendix I

Project Poster

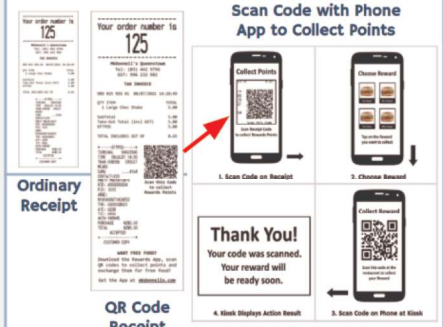
MG7101 - Engineering Project
Stephen Julian (ID: 1041664)
https://linkedin.com/in/nzstevejulian/

QR Code Scanning Kiosk

BEngTech (Electrical)
Department of Electrotechnology
Unitec Institute of Technology

1 Background

Customer Loyalty Programme



2 Methodology

Research & Development Processes

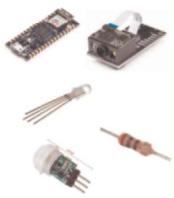
Develop a series of rapid incremental prototypes, each with features and functions that improve on the previous prototype.

Version 1
Learn ▼ Improve
Version 2
Learn ▼ Improve
Version 3
Learn ▼ Improve
Version 4
Learn ▼ Improve
Version 5

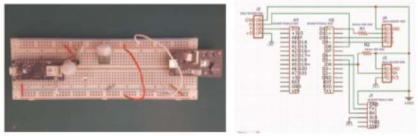
3 Prototype V1

Breadboard Circuit

1. Arduino Nano IoT 33 MCU
2. QR Code Scanner Module
3. PIR Motion Sensor
4. RGB LED Status Indicator
5. Miscellaneous Components



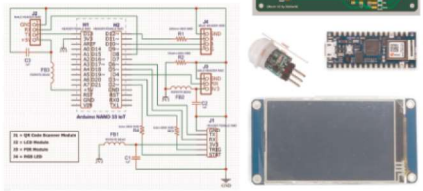
Functional Testing and Proof of Concept



4 Prototype V2

Custom PCB V1

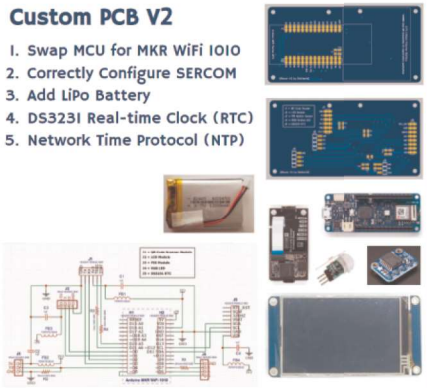
1. Arduino Nano IoT 33 MCU
2. Add Custom PCB for Modules
3. Add LCD Display Module
4. Add Decoupling Capacitors
5. Add Ferrite Beads as Filters



5 Prototype V3

Custom PCB V2

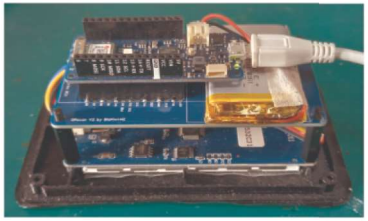
1. Swap MCU for MKR WiFi 1010
2. Correctly Configure SERCOM
3. Add LiPo Battery
4. DS3231 Real-time Clock (RTC)
5. Network Time Protocol (NTP)



6 Prototype V4

Secure Server Connectivity

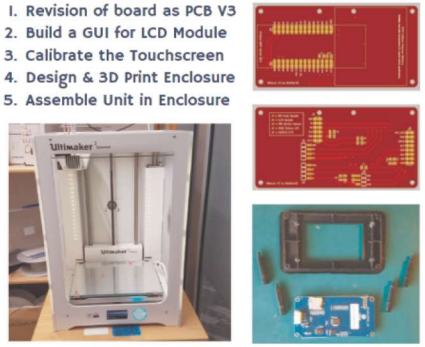
1. Add CSR Requests, x.509 and CA Certificates
2. Store Keys in One Time Locked Crypto Hardware
3. Add Arduino BearSSL
4. Add Message Queuing Telemetry Transport (MQTT)
5. Amazon IoT Core: Things, Topics, Policies, Certs



7 Prototype V5

Final Touches, Stretch Goals, Future

1. Revision of board as PCB V3
2. Build a GUI for LCD Module
3. Calibrate the Touchscreen
4. Design & 3D Print Enclosure
5. Assemble Unit in Enclosure



8 Conclusions

Prototypes Outcomes

Version 1 = Completed.
▼
Version 2 = Completed.
▼
Version 3 = Completed.
▼
Version 4 = Completed.
▼
Version 5 = Initiated, Stretch Goals for Future

Opportunities for Improvement

1. USB Connectivity
2. Ethernet and PoE Connectivity
3. Bluetooth Connectivity
4. NFC Interface
5. Single PCB Board
6. Certifications

Appendix J

Modern Tools Usage by Stephen Julian during MG7101 project paper:

	Application / Purpose / Benefit
Software Tools	
- Fritzing 0.9.6 beta	Circuit prototype design tool.
- Paint 3D	Viewing 3D STIL files.
- Arduino IDE 1.8.15	Uploading of firmware to microcontroller.
Web Services	
- EasyEDA (easyeda.com)	Electronic design of Printed Circuit Boards (PCB). Order placement for manufacture of batches of PCB's.
- Google Apps for Business	Communication, Documentation and file storage.
- Circuit Lab (circuitlab.com)	Circuit schematic builder.
Hardware Tools	
- Laptop PC	Software development, PCB design, device testing, documentation.
- Smartphone	Capturing photos of device fabrication, assembly and testing.
- Multimeter	Circuit continuity testing (e.g. of PCB's received from the manufacturer)
- USB to TTL cable adaptor	Cable adapter for interfacing host PC with Nextion LCD module.
- Thermostatic Soldering Iron, Clamp and Accessories	Soldering of SMD components on to prototype PCBs.
- Drill bits	Enlarging holes for standoffs in PCB's.
Consumables	
- Lead-Free Solder	Soldering of prototypes.

Appendix K

Graduate Attributes

Note: Attribute names & Definitions are sourced from Unitec MG7101 course content.

Attributes	Definitions and Evidences of Achievement
Engineering Knowledge	Apply knowledge of mathematics, natural science, engineering fundamentals and an engineering specialization to defined and applied engineering procedures, processes, systems or methodologies. ***Evidences of Achievement:*** • Completed: NZDE (Electronics) and BEngTech (Electrical) (2021). • NZDE Project: DE6102 Project. • BEngTech Project: MG7101 Project. • Completion of four tertiary level mathematics courses. • Completion of two tertiary level Engineering Mechanics courses. • Specialization: Electronics, Microcontrollers, Embedded Systems. • Industry Employment: Roles involving Electrotechnology.
Problem Analysis	Identify, formulate, research literature and analyse *broadly-defined* engineering problems reaching substantiated conclusions using analytical tools appropriate to the discipline or area of specialisation. ***Evidences of Achievement:*** • BEngTech Project: MG7101 Project. • Electronics Design Project: MG6024 Electronics Design. • Industry Employment: Project roles in Electrotechnology and ICT.
Design & Development of Solutions	Design solutions for *broadly- defined* engineering technology problems and *contribute to* the design of systems, components or processes to meet specified needs with appropriate consideration for public health and safety, cultural, societal, and environmental considerations. ***Evidences of Achievement:*** • BEngTech Project: MG7101 Project. • Electronics Design Project: MG6024 Electronics Design. • Industry Employment: Roles involving Electrotechnology and ICT.
Investigation	Conduct investigations of *broadly-defined* problems; locate, search and select relevant data from codes, databases and literature, design and conduct experiments to provide valid conclusions. ***Evidences of Achievement:*** • BEngTech Project: MG7101 Project. • Industry Employment: Roles involving Electrotechnology and ICT.

Modern Tool Usage	Select and apply appropriate techniques, resources, and modern engineering and IT tools, including prediction and modelling, to *broadly- defined* engineering problems, with an understanding of the limitations. ***Evidences of Achievement:*** • Mathematics: Use of MATLAB and SciLab simulation tools. • Electrical: Use of Multimeter, Oscilloscope, Signal Generator. • Electronics: Use of MultiSim circuit simulation tools. • Microcontrollers: Use of Atmel Studio, Arduino IDE tools. • Embedded: Use of XiLinx Vivado for FPGA development. • Electronics Manufacturing: Use of EDA tools. • NZDE Project: DE6102 Project. • BEngTech Project: MG7101 Project. • Industry Employment: Roles involving Electrotechnology and ICT. • **Refer to Appendix K for more specific examples of evidence.**
The Engineer and Society	Demonstrate understanding of the societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to engineering technology practice and solutions to broadly defined engineering problems. ***Evidences of Achievement:*** • NZDE Course: DE6101 Engineering Management Principles. • BEngTech Course: MG7121 Professional Engineering Practice. • Membership: Engineering New Zealand. • Membership: Electrical Engineers Association (EEA). • Industry Employment: Roles involving Electrotechnology and ICT.
Environment and Sustainability	Understand and evaluate the sustainability and impact of engineering technology work in the solution of broadly defined engineering problems in societal and environmental contexts. ***Evidences of Achievement:*** • NZDE Course: DE6101 Engineering Management Principles. • BEngTech Course: MG7121 Professional Engineering Practice. • Membership: Engineering New Zealand. • Membership: Electrical Engineers Association (EEA). • Industry Employment: Roles involving Electrotechnology and ICT.
Lifelong Learning	Recognize the need for, and have the ability to engage in independent and life-long learning in specialist technologies. ***Evidences of Achievement:*** • Continuous enrolment in tertiary education since 2012. • Four industry recognised qualifications over the last 10 years. • Over 20 years of employment in technology focused roles.

Ethics	Understand and commit to professional ethics and responsibilities and norms of engineering technology practice. ***Evidences of Achievement:*** • NZDE Course: DE6101 Engineering Management Principles. • BEngTech Course: MG7121 Professional Engineering Practice. • Membership: Engineering New Zealand. • Membership: Electrical Engineers Association (EEA). • Industry Employment: Roles involving Electrotechnology and ICT.
Individual and Team Work	Function effectively as an individual, and as a member or leader in diverse teams. ***Evidences of Achievement:*** • NZDE Course: DE6101 Engineering Management Principles. • NZDE Project: DE6102 Engineering Project. • BEngTech Project: MG7101 Engineering Development Project. • BInfoTech Project: IT701 Project (Project Manager). • Industry Employment: Roles involving Electrotechnology and ICT.
Communication	Communicate effectively on *broadly-defined* engineering activities with the engineering community and with society at large, by being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions. ***Evidences of Achievement:*** • NZDE Course: DE4103 Technical Literacy. • NZDE Course: DE6101 Engineering Management Principles. • NZDE Project: DE6102 Engineering Project. • BEngTech Course: MG7121 Professional Engineering Practice. • BEngTech Project: MG7101 Engineering Development Project. • BInfoTech Project: IT701 Project (Project Manager). • Slide and Video presentations, Report writing, Poster design. • Industry Employment: Roles involving Electrotechnology and ICT.
Project Management and Finance	Demonstrate knowledge and understanding of engineering management principles and apply these to one's own work, as a member or leader in a team and to manage projects in multidisciplinary environments. ***Evidences of Achievement:*** • NZDE Course: DE6101 Engineering Management Principles. • NZDE Project: DE6102 Engineering Project. • BEngTech Project: MG7101 Engineering Development Project. • BInfoTech Project: IT701 Project (Project Manager). • Industry Employment: Project roles in Electrotechnology and ICT.

Appendix L

The pages that follow in this Appendix feature:

(1.) Cover Letter from Stephen Julian, followed by

(2.) Curriculum Vitae (CV) of Stephen Julian.

Stephen Julian

BInfoTech • NCEE • NZDE (Electronics) • BEngTech (Electrical, 2021)

ADDRESS: 59 Thompsons Track, RD 2, Katikati 3178, Bay of Plenty, NEW ZEALAND

T: 022 360 1234 • E: steve.julian@orcon.net.nz • L: https://linkedin.com/in/nzstevejulian/

29 August 2022

Dear Miles,

I am writing to you to inquire about open positions at your esteemed organization.
I learned of **Cyberdyne Systems Limited** through a robotics workshop I participated in and I found your advances in research and development very interesting.
I'm passionate about Electrotechnology and would like to know if you have any upcoming employment openings in your Special Projects Division?

I believe my skills and qualifications will be suited for your company's needs. I have four industry recognised qualifications in Electrical Engineering and Information and Communications Technology and 20 years of experience working in digital and electrotechnology focused roles.

Please find enclosed my Curriculum Vitae for your consideration. I will be enthusiastic about any opportunity to discuss any potential employment, contracting or consulting opportunities for working with your company.
You can reach me via my mobile phone at 022 360 1234 at a time convenient for you.

Thank you for your time and consideration. I look forward to discussing future opportunities with you soon.

Kind regards,

Stephen Julian

Stephen Julian - Curriculum Vitae

BInfoTech • NCEE • NZDE (Electronics) • BEngTech (Electrical, 2021)

ADDRESS: 59 Thompsons Track, RD 2, Katikati 3178, Bay of Plenty, NEW ZEALAND

T: 022 360 1234 • E: steve.julian@orcon.net.nz • L: https://linkedin.com/in/nzstevejulian/

Objective

To develop and apply my skills in software, firmware and hardware as an Electrotechnologist.

Personal Statement

I'm a highly motivated team player looking for new opportunities in the technology industry.
My background is in software development which I'm now diversifying into electronics hardware and firmware through working in the emerging field of Internet-of-Things (IoT).
I have four industry recognised qualifications in Electrical Engineering and Information Technology.
My industry skills, knowledge, strengths, training and work experience are extensive and summarised in these pages and on my LinkedIn profile.
Technical challenges are my forté and with these I expect to be a valued asset in a Team.
We are limited only by what we believe to be possible and I'm inspired to prove this in my work.
For further information please feel free to contact me by any of the methods at top of this page.

Industry Skills & Knowledge

Industry skills and knowledge in Electronics Engineering and Information Technology include:

Engineering Knowledge	**Problem Analysis**	**Engineering Solutions**	**Investigation**
Modern Tool Usage	**The Engineer and Society**	**Environment & Sustainability**	**Ethics**
Individual & Teamwork	**Communication**	**Project Management**	**Lifelong Learning**

Technical Strengths

Technical strengths include hardware, firmware, software, server systems and telecommunications:

Software Development	**Internet of Things**	**Electronics**	**Server Administration**
Agile & Waterfall	**Single Board Computers**	**Electronics Assembly**	**Linux / VPS / Cloud**
Technical Writing	**Microcontrollers**	**Circuit Schematics**	**CLI / BASH / Shell / Batch**
Source Code Management	**Firmware & Embedded**	**EDA Tools for PCB Design**	**ICT Security**
Full Stack Development	**RTOS**	**Sensors & Actuators**	**Data Networking**
REST API	**FPGA / Verilog**	**Barcode / UPC / QR Code**	**Telecommunications**
Databases	**Rapid Prototyping**	**RFID & NFC**	**Simulation Tools**

Employment History

Hardware & Software Systems Engineer (Sep 2019 - Mar 2020 and Aug 2021 - Present)

Where: **BizKiwi NZ** (Katikati, Auckland)

Scope: Contracted services building concept prototype monitoring device for a medical device manufacturer.
Microcontroller firmware and server development for monitoring sensors and driving actuators.
Microcontroller, Arduino, Firmware, Sensors, Actuators, Sigfox ThinXtra, OG, 2G, 3G, Internet of Things (IoT), Cryptography, MQTT, AWS IoT Core, AWS DynamoDB.

Ecommerce Solutions Developer (Mar 2010 - Present)

Where: **BizKiwi NZ** (Katikati, Auckland, Queenstown, Invercargill)
BizKiwi is my brand through which I provide contracted technical services.

Scope: Contracted services building B2B and B2C websites for the Shopify ecommerce platform.
Customisation and optimisation of Shopify themes for security, speed and enhanced conversion ratios.
Search Engine Optimisation (SEO), Google Analytics, Google Search Console, Lighthouse, Permission Marketing, Blog, Social Media, Facebook Marketing, Google Adwords, Google MyBusiness, StockSync FTP Feed, Google Sheets, Database, Product Images, UI/UX, Liquid, HTML/CSS/JS, REST.

Other: *Branding, Logos, Business Cards, Stationery, Flyers, Banners, Vehicle Signage, QR codes.*

Hardware & Software Systems Engineer (Dec 2014 - June 2018)

Where: **BizKiwi NZ** (Auckland, Queenstown, Invercargill)

Scope: Contracted services building QMS and order tracking systems for the Restaurant & Hospitality industry. Development of software, firmware, hardware and hosted server systems.
Order Ready Management, Product Timer, QMS, Build Training System, Environment Sensor, Touchscreen, Embedded, Web, Mobile, Network, Cloud, Wireless, Real-time Monitoring, RS-232, Laravel, REST API.

Contract Developer and Technician (June 2008 - Mar 2010)

Where: **Self-Employed** (Auckland, Queenstown, Invercargill)

Scope: Technical Consultant for the Digital Imaging, Creative and Hospitality industries.
Activities included installation of wired and wireless network infrastructure as well as PoE-based CCTV monitoring systems in restaurant environments.

Web Developer & Technical Trainer (Apr 2006 - June 2008)

Where: **MediaMind Group** (Auckland, New Zealand)

Scope: Web Developer & Technical Training Consultant working with *HTML/JS/CSS, ASP.NET, MS SQL Server.*

Software Developer & Technical Consultant (Mar 2002 - Apr 2006)

Where: **Fish Logic Limited** (Auckland, New Zealand)

Scope: Service provision for the Electronics, Digital Imaging and E-commerce industries.
Activities included systems integration, electronic device assembly and testing.

Other: *Design, branding, logos, business cards, stationery, brochures, reports, banners, promo media.*

Academic History

I have four industry recognised qualifications in Electrical Engineering and Information Technology:

Bachelor of Engineering Technology - BEngTech (Electrical) Major: Electronics

Where: **Unitec Institute of Technology** (Auckland, New Zealand)

When: July 2018 to Nov 2021 **(finalising, 23 of 24 papers completed, 12 are credited from NZDE)**

Major: Electronics, Microcontrollers, FPGA, PCB Design, CAD Design, Internet of Things (Level 7).

Other: Electronics Design Project (Wireless Sensor Monitoring Device - design and build).

MATLAB, Scilab, MultiSim, Autodesk SolidWorks, Autodesk AutoCAD, Atmel Studio.

New Zealand Diploma of Engineering - NZDE (Electronics)

Where: **Unitec Institute of Technology** (Auckland, New Zealand)

When: July 2016 to 2018 **(Graduated)**

Major: Electronics, Microcontrollers and PCB Design for Manufacture (Levels 4, 5 and 6).

Other: Project: RS-232 Serial Monitor - real-time data packet interception, analysis, events).

National Certificate of Electrical Engineering - NCEE

Where: **Southern Institute of Technology** (Invercargill, New Zealand)

When: Feb 2016 to June 2016 **(Graduated)**

Major: Electrical Engineering Unit Standards (Levels 2 and 3) recognised by NZQA and EWRB.

Other: **Certificate in Intermediate First Aid** (6400, 6401, 6402).

Certificate in Hand Operated Fire Fighting Equipment (Practical & Theory).

Engineering Trades Scholarship recipient awarded by Invercargill Licensing Trust.

Bachelor of Information Technology - BInfoTech, Major: Software Development

Where: **Southern Institute of Technology** (Queenstown and Invercargill, New Zealand)

When: Feb 2012 to Nov 2015 **(Graduated)**

Major: Internet-of-Things (IoT), Full Stack Development, Software Engineering (Levels 5, 6, 7)

Other: Double Project: Interactive 360 degree Digital Imaging System

Mobile Application Development (Post-Grad. Level 8)

Web Application Development (Post-Grad. Level 8)

Student Peer Tutoring (employed by SIT)

W3Schools Training (HTML/CSS/JS)

3D modeling with Autodesk Maya.

Extramural Training

2013 : Silverstripe Framework and CMS (Southern Institute of Technology)

2012 : WordPress CMS (Southern Institute of Technology)

2012 : Achievo ATK Framework (Southern Institute of Technology)

2012 : PHP / MySQL (Southern Institute of Technology)

Memberships

2021+ : Electricity Engineers Association (EEA) - Student Member

2018+ : Engineering New Zealand (formerly IPENZ) - Student Member

References

Referees, References and Academic records are available on Request.