|  |  |
| --- | --- |
| CURRENT TRASFORMER METER  CTM-3501 | Abstract  This report contains the design, implementation, and testing of a non-invasive near to real-time current metering protocol for alternating systems.  Michael Giorgas & Clinton Elliott  Electrical Engineering BA (Electronic) |

You had me at “Hello World”;

-Anonymous

TABLE OF CONTENTS

[1. INTRODUCTION 1](#_Toc496114166)

[2. OVERVIEW OF DESIGN 2](#_Toc496114167)

[3. HARDWARE DESIGN 3](#_Toc496114168)

[3.1. Schematic 3](#_Toc496114169)

[3.1.1. Power Supply 3](#_Toc496114170)

[3.1.2. Operational Input Shifters 3](#_Toc496114171)

[3.1.3. Multiplexer 4](#_Toc496114172)

[3.1.4. ZigBee 5](#_Toc496114173)

[3.1.5. M5 & H5 Processors 5](#_Toc496114174)

[3.1.6. Mini-USB 6](#_Toc496114175)

[3.1.7. JTAG and SDA Headers 7](#_Toc496114176)

[3.1.8. Battery Sizing & Maximum Current (MICK) 7](#_Toc496114177)

[3.1.9. 3-Stage Contingency 8](#_Toc496114178)

[3.1.1. Power Saving 8](#_Toc496114179)

[3.2. PCB Design 9](#_Toc496114180)

[4. SOFTWARE DESIGN 11](#_Toc496114181)

[4.1. Kinetis Code 11](#_Toc496114182)

[4.1.1. Analog to Digital Converter 11](#_Toc496114183)

[4.1.2. Timing 11](#_Toc496114184)

[4.1.3. Vectorization 12](#_Toc496114185)

[4.1.4. Timers on Interrupts 12](#_Toc496114186)

[4.1.5. Waiting for Interrupts 12](#_Toc496114187)

[4.1.6. ZigBee Transmission (MICK) 12](#_Toc496114188)

[4.2. C++ Code 12](#_Toc496114189)

[4.2.1. ZigBee Data Acquisition (MICK) 12](#_Toc496114190)

[4.2.2. WiPi Upload (MICK) 12](#_Toc496114191)

[4.2.3. ThingSpeak Display (MICK) 12](#_Toc496114192)

[5. TESTING & DISCUSSION 13](#_Toc496114193)

[5.1. Game-Day Performance 13](#_Toc496114194)

[6. LIMITATIONS & RECOMMENDATIONS 14](#_Toc496114195)

[6.1. Limitations 14](#_Toc496114196)

[6.2. Recommendations 14](#_Toc496114197)

[7. CONCLUSION 15](#_Toc496114198)

[8. REFERENCES 16](#_Toc496114199)

[APPENDIX A – SCHEMATIC 1](#_Toc496114200)

[APPENDIX B – PCB LAYOUT 2](#_Toc496114201)

[APPENDIX C – BILL OF MATERIALS 3](#_Toc496114202)

[APPENDIX C – CODE 4](#_Toc496114203)

LIST OF FIGURES

[Figure 1 - Current Transformer [1] 1](file:///C:\Users\Clint\Downloads\CT-Metering-Assignment-R_2.docx#_Toc496114065)

[Figure 2 -Overview of Operation 2](#_Toc496114066)

[Figure 3 - Power Supply and Corresponding Protection 3](file:///C:\Users\Clint\Downloads\CT-Metering-Assignment-R_2.docx#_Toc496114067)

[Figure 4 - Operational Amplifiers and Input Shifting Circuit 4](file:///C:\Users\Clint\Downloads\CT-Metering-Assignment-R_2.docx#_Toc496114068)

[Figure 5 - Multiplexer 4](file:///C:\Users\Clint\Downloads\CT-Metering-Assignment-R_2.docx#_Toc496114069)

[Figure 6 - ZigBee Transmitting Device 5](file:///C:\Users\Clint\Downloads\CT-Metering-Assignment-R_2.docx#_Toc496114070)

[Figure 7 - Primary (left) and SDA (right) Processors and Oscillators (bottom right) 6](file:///C:\Users\Clint\Downloads\CT-Metering-Assignment-R_2.docx#_Toc496114071)

[Figure 8 - Mini USB Circuit 6](file:///C:\Users\Clint\Downloads\CT-Metering-Assignment-R_2.docx#_Toc496114072)

[Figure 9 – SDA (left) and JTAG (right) Headers 7](file:///C:\Users\Clint\Downloads\CT-Metering-Assignment-R_2.docx#_Toc496114073)

[Figure 10 - Coin Cell Battery and Isolator 7](file:///C:\Users\Clint\Downloads\CT-Metering-Assignment-R_2.docx#_Toc496114074)

[Figure 11 – FRDM Board Headers 8](file:///C:\Users\Clint\Downloads\CT-Metering-Assignment-R_2.docx#_Toc496114075)

[Figure 12 - Auto Routed Test of Placement 9](file:///C:\Users\Clint\Downloads\CT-Metering-Assignment-R_2.docx#_Toc496114076)

[Figure 13 - SDA Processor with Poor Routing 9](file:///C:\Users\Clint\Downloads\CT-Metering-Assignment-R_2.docx#_Toc496114077)

[Figure 14 - Finished PCB 10](file:///C:\Users\Clint\Downloads\CT-Metering-Assignment-R_2.docx#_Toc496114078)

[Figure 15 - Definition of Timing in Circuit (seconds) 11](file:///C:\Users\Clint\Downloads\CT-Metering-Assignment-R_2.docx#_Toc496114079)

CURRENT TRASFORMER METER

# INTRODUCTION

The aim of this project was to design and implement the CTM-3501. This report outlines the design of the circuit with calculations and diagrams. The implementation in Code Warrior using Free Scale will be discussed and explained. The final product and testing will be evaluated, commented upon, and recommendations are given.

The CTM-3501 (CTM for short) stands for Current Transformer Meter and followed by the model number i.e. CTM-xxxx. The 35xx series are the top of the range on the market. The CTM was initially designed to meet the minimum requirements of having an embedded system with a sensor which acquires data. The features to be implemented were given flexible conditions, while the team behind the CTM were driven to achieve a glorious result.

A current transformer (CT) is and electrical device used to step or scale current down to a safe working current to be measured by meters and relay for protection and monitoring. The CT gives a proportional current in its secondary winding to the current in the primary winding based on the turns ratio, and should have approximately a negligible load. Owing to the construction of CTs, they provide electrical isolation from the high voltage circuit through the windings, consequently increasing safety. Generally, these transformers have a small number of primary turns and a larger number of secondary turns which defines the turns ratio [1]. Figure 1 shows a VT comprising of the primary and secondary windings, and earthed secondary winding.

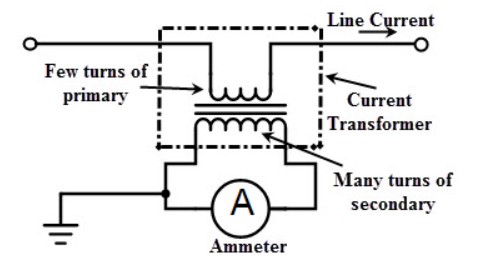


Figure 1 - Current Transformer [1]

There are several types of CTs including wound type, bar type and window type transformers. The CT behaves similar to a normal transformer in no load conditions (small burden) and therefore the secondary normally has low currents and is earthed for safety precautions. The current on the secondary winding is governed by the equation:

Where is the primary winding current, is the secondary winding current, and are the number of turns for the primary and secondary windings respectively. If the secondary winding is not shorted (by a burden resistor) than high dangerous voltages can occur governed by equation:

To demonstrate this effect, we assume a and are 1 and 100 with a primary voltage of 230. Substituting into the equation:

Therefore, a CT with a turns ratio of 100:1 can develop 23 kV with nominal phase voltage. Most insulation is rated to 1000 V and this high voltage can degrade the insulation creating hazards. Current transformers are used in many aspects of a power system. They are utilized in power distribution, generating stations, substations and at domestic, commercial and industrial levels [1].

The essence of the design was to monitor an alternating current using a current transformer (CT). The CT’s were to be part of a sensor node which sent information via wireless communication to a base station. The base station was a Raspberry Pi which could forward near to real-time data upload to the internet. The complete design process of the CTM is explained in this document and a corresponding user manual is attached.

# OVERVIEW OF DESIGN

The fundamental operation of the CTM is displayed in Figure 2 and depicts the start of the data acquisition from the current transformers. The voltage is shifted using the wave shaping and converted to a digital signal. This signal is then computed to get a desired output based on the calibration phase and type of current transformer being used. The data is then sent over radio frequency to a base station Raspberry Pi and uploaded via WiPi to ThingSpeak for webpage display.

Figure 2 -Overview of Operation

# HARDWARE DESIGN

## Schematic

The CTM was based on the embedded system design of the Kinetis FRDM K20DX128M5 board for the processing circuit. The schematic was compartmentalized into several parts during the design process and each part is discussed as per its design in the following sections.

### Power Supply

The two processors each required 3.3 V power supply, so during the design this was the primary reason for selecting 3.3 V. The power design is shown in Figure 3.

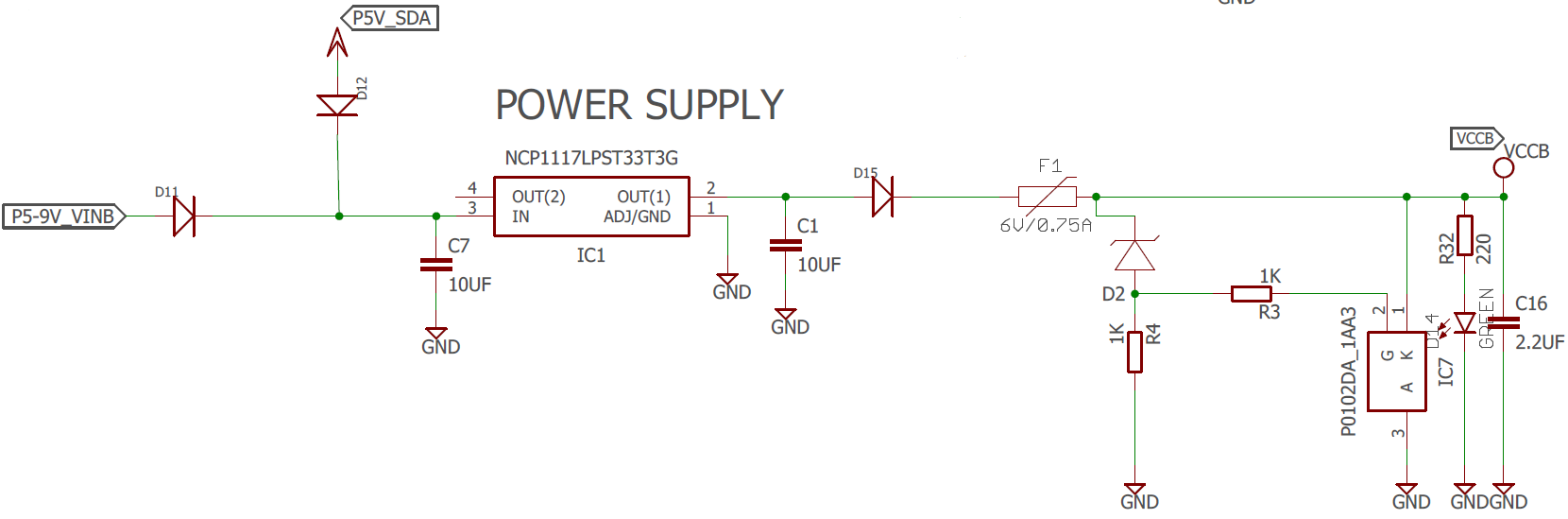


Figure 3 - Power Supply and Corresponding Protection

The supply to the circuit was from a 9 V coin-cell battery or from a 5 V USB with each passing through the diodes D11 and D12 to prevent incorrect connection. Each supply was fed into a power supply with the capacitor sizes selected from the data sheet for the NCP1117LPST33TCG. It was noted that at 3.3 V the regulator could output current in excess of 1.0A. The poly fuse F1 was selected at 0.75 A as this would protect the regulator and was above the calculated maximum current draw of the circuit (See Battery Sizing & Maximum Current). The SCR and D2 and resistor provide Crowbar protection in the event of voltage spikes. D2 was selected at 3.5 V to prevent transients entering the processors and causing damage. The poly fuse provides over-current protection and the crowbar protection provides over-voltage protection.

### Operational Input Shifters

The CTs transduce the current into a voltage which is fed into the LM358 op-amp. The output voltage is level shifted from the input bias voltage on the non-inverting pin from the voltage division over the potentiometer VPOT as shown in Figure 4.

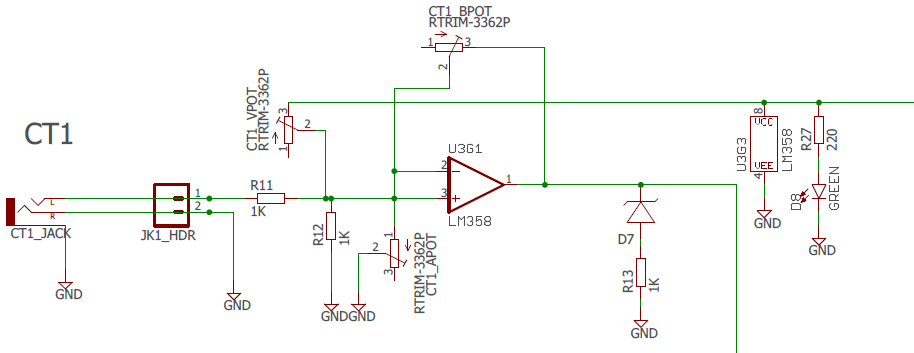


Figure 4 - Operational Amplifiers and Input Shifting Circuit

Figure 4 shows the output is clipped by the diode D7 which was set at 3.3 V. If the output voltage was above this value then it was clipped to earth protecting the input pins to the processor against transients. The original values for the POTs were calculated but it was found that each need to be adjusted due to manufacturer tolerances. The wave form need to be shifted positively half the maximum input voltage of 3.3 V. This meant 1.65 V input bias but this was affected during prototyping due to power supply voltages. Note the headers which were installed near the headphone jack for testing purposes.

### Multiplexer

The multiplexer (MUX) was incorporated to reduce the power consumed by the op-amps and POTs. The MUX was supplied power by the microprocessor and switched between the op-amp circuits. This allowed the op-amp circuits to be only powered while data was being recorded. The MUX is shown in Figure 5 and the timing is shown in Figure 15.

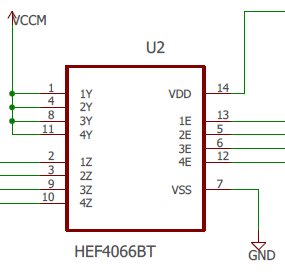


Figure 5 - Multiplexer

The MUX is powered during the cycle (see Figure 15) and switches between CT1, CT2, CT3 and CT4. This allows only one CT circuit to be powered at a time reducing the power consumed by four. The upshot of switching from the microprocessor is that the MUX is only power during the cycle which reduces the amount of power consumed by the MUX by 15.

### ZigBee

The Xbee module brand a Zigbee compliant radios made by Digi International wwas chosen for the wireless communication due to its low power, low data rate which is ideal for battery applications. The Zigbee communication standard supports mesh networks and is used in applications with sensor networks which require machine-to-machine communications.

The transmitter used was the ZigBee. This was selected due to familiarity and the ability to place the device into sleep mode by switching the sleep pin from the microprocessor and is shown in Figure 6. This meant that only once a phase the ZigBee had to be powered which decreased the power consumption.

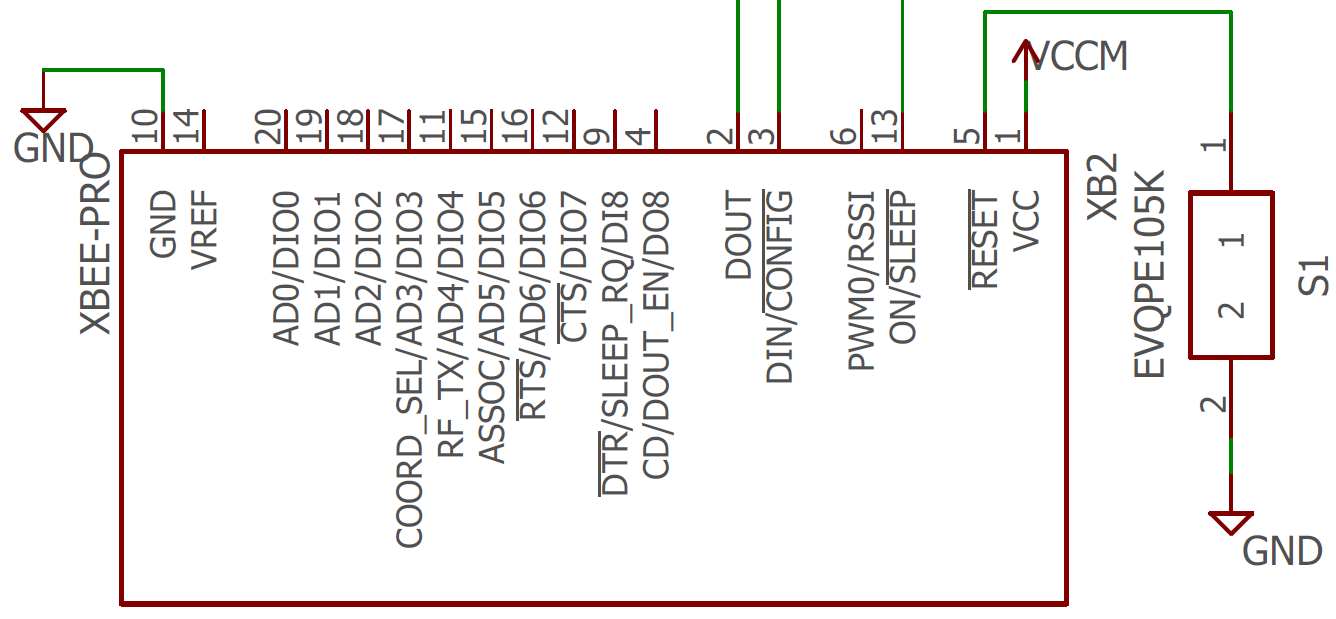


Figure 6 - ZigBee Transmitting Device

### M5 & H5 Processors

There were two microcontrollers used in the design: the MK20DX128VLH5 and the MK20DX128VFM5 which will be referred to here after as the Primary and the SDA processors. As mentioned previously these two processors were selected as they were based off the FRDM board and are shown in Figure 7. These two processors required three oscillators shown in Figure 7 at frequencies of 32 kHz and 8 MHz.

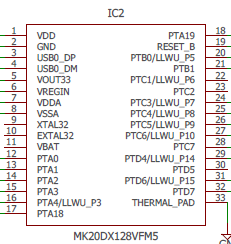
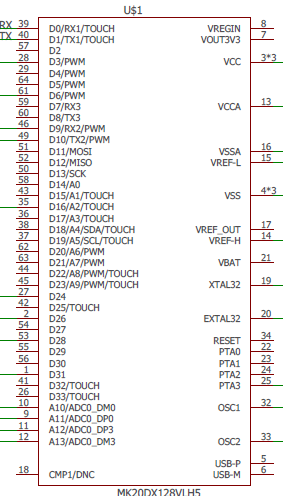
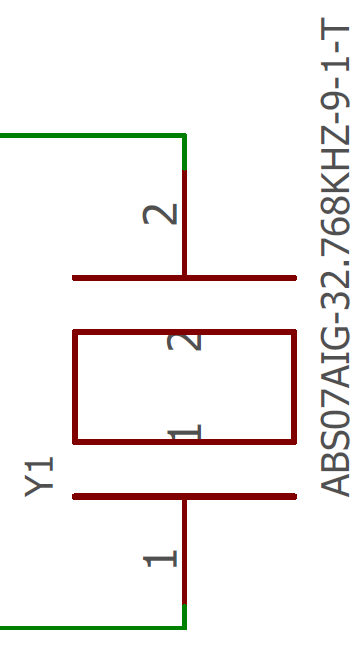
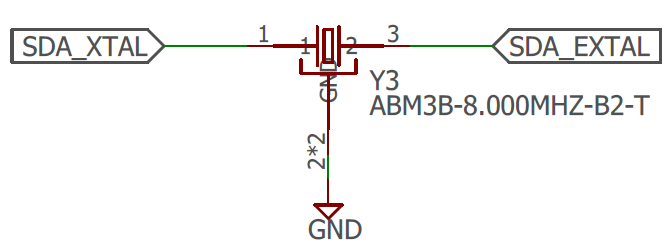


Figure 7 - Primary (left) and SDA (right) Processors and Oscillators (bottom right)



### Mini-USB

The mini USB (Figure 8) was selected to allow connection to the SDA processor to convert the signals to serial to program the primary processor. The BZT52C15S Zener (D1) was installed to protect the board from high input transients and static. The L1 and L2 inductors or ferrite beads are installed to further decrease these effects.

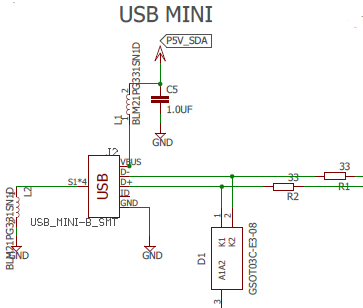


Figure 8 - Mini USB Circuit

### JTAG and SDA Headers

The JTAG and SDA headers were installed to allow serial communication directly to both processors and are shown in Figure 9. The SDA header allowed communication to the primary processor and the JTAG header allowed communication to the SDA processor.

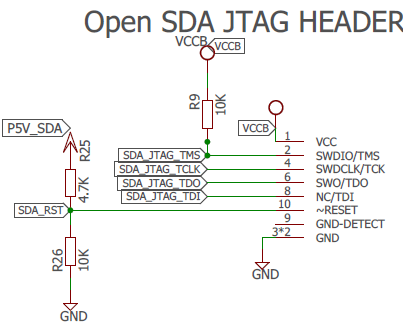
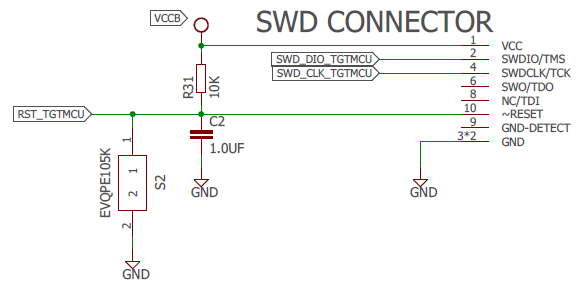


Figure 9 – SDA (left) and JTAG (right) Headers

### 3-Stage Contingency

During the development of the CTM system it was decided to include stages on contingency due to the frontiers which were being pushed.

* Stage 1 Contingency was to essentially build the board in reference to headers which could be connected to a FRDM board. This meant the CTM could be inserted into a FRDM board and operate. This was achieved by connecting to FRDM headers (see Figure 11) all the input and outputs from the CTs. The CT circuits would receive power from the FRDM board and this supply rails was called the Backup (VCCB)

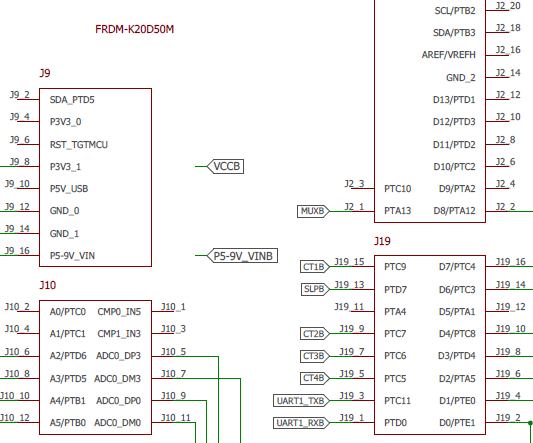


Figure 11 – FRDM Board Headers

* Stage 2 Contingency was having the primary processor drive the circuit connected across the FRDM headers by bridges. This circuit was powered by the CTM power supply and labelled the Main power supply (VCCM).
* Stage 3 Contingency was including the SDA chip and allowing the CTM to be programmed by mini USB interface.

### Battery Sizing & Maximum Current (MICK)

The battery was isolated from the board by a 2-position dip switch which disconnected the positive and negative of the battery.

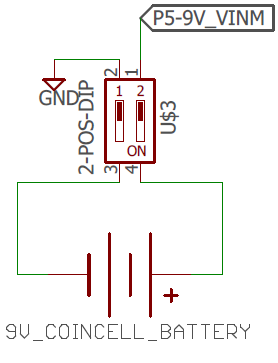


Figure 10 - Coin Cell Battery and Isolator

All electronic systems require power supplies to operate, Power management broadly refers to the generation and control of regulated voltages required to operate an electronic system. It encompasses much more than just power supply design. Today's systems require power supply design be integrated with the system design to maintain high efficiency

Integrated circuit components such as switching regulators, linear regulators, switched capacitor voltage converters, and voltage references are typical elements of power management.

The battery life of the circuit is determined by the battery capacity, which is the amount of current in total that the battery can supply during the lifetime of the battery and the load current of the circuit, the amount of current being consumed by the circuit. The first factor can be found by the manufacturer details of the battery which gives the capacity in amp-hours or milliamp- hours, the second factor requires to calculate the current drawn from the active devices in the circuit. The following formula is used to determine the battery life.

#### Battery Capacity

The Energizer 522 Alkaline 9-volt battery capacity is rated at 230 milliamp-hours at a 1000 milliamp discharge rate.

### Power Saving

#### Current Drawn

*Operational Amplifiers*

To determine the power draw of the op amp the quiescent current was obtained from the data sheet and the following formula used:

The data sheet provided a typical value and a maximum value for the quiescent current, each value was used to obtain the power of the op amp with the max value used as a worst-case scenario.

per Op-Amp

per Op-Amp

The component is a dual Op-Amp package therefore each package will draw and the board has 4 in total resulting in

*Voltage Regulator*

The maximum package power dissipation of the NCP1117LP voltage regulator is given by the formula:

is the maximum junction temperature range

is the operating ambient temperature range

is the thermal Resistance, Junction−to−Ambient

The maximum power

The maximum current output of the regulator is where V is the voltage drop of the regulator.

To calculate a typical power rating of the voltage regulator it is somewhat hidden within the datasheet. It can be calculated by looking at the following related specifications.

is the operating junction temperature range

is the thermal resistance junction-to-case

is the thermal resistance, junction−to−ambient

specifies how hot the “junction”, the active part of the regulator can get before it goes into thermal shutdown. specifies how much temperature difference to be expected between the junction and the outside of the package. This is relevant if you can quickly remove heat from the package. A perfect coupled heat sink hooked to your package, for each Watt the junction temperature would rise only above the temp of the heat sink. is how hot the junction gets when the regulator is dissipating a given amount of power and the regulator is sitting at a given ambient temperature. We design our regulator to work under modest commercial conditions such that it will not exceed and the junction temperature needs to stay below then the maximum temperature rise allowable is The power dissipation is given by:

The current is

*XBee Pro*

The XBee Pro attached to the board will draw current during transmission and when in its sleep mode. The maximum current draw will be transmission mode with the data sheet quoting a transmit current of The power down current of the XBee is

*MK20DX128VLH5 and MK20DX128VFM5 processors.*

The absolute maximum ratings for the MK20DX128VFM5 and MK20DX128VLH5 are obtained from the device datasheet. The maximum power supply current , includes all current being sourced by the microcontroller pins in addition to the current used to operate the CPU and peripherals. For the MK20DX128VLH5 and MK20DX128VFM5 the

*SN74LVC125A Quadruple Bus Buffer Gate*

The absolute maximum current draw of the SN74LVC125A was calculated from the manufacturers datasheet by adding the continuous input current and the output current.

*HEF4066B Quad single-pole single-throw analog switch(Multiplexer)*

The max supply current at worst case scenario for the Multiplexer was when the device is operating at an ambient temperature of .

*Mini USB*

The specifications of USB 2.0 states that the maximum current draw is .

*LED’s*

The current drawn from the LED’s can be found using the formula:

Supply voltage

LED forward voltage drop (found in datasheet)

Resistor value

The board has a total of 6 LED’s taken the worst case that all LED’s are on at the same time the total current draw

The table below shows the total maximum current for each component. These values are a worst-case scenario in which the absolute maximum current would be supplied to each component. The battery life calculation will be done using the worst-case scenario for each component, assuming that all components are at maximum draw at the same time and operating all the time. The total is , which is approximately 1000mA.

U ing the battery life equation:

|  |  |
| --- | --- |
| Component | Maximum Current |
| Op Amps |  |
| Voltage Regulator |  |
| XBee Pro |  |
| MK20DX128VLH5 |  |
| MK20DX128VFM5 |  |
| Quadruple Bus Buffer Gate |  |
| Quad single-pole single-throw analog switch(Multiplexer) |  |
| LED’s |  |
| Mini USB |  |
|  |  |

## PCB Design

The selected method of layout approach for the PCB was trial and error. This approach allowed many designs to be made, consuming copious amounts and time, and delivering maximum returns in the form of stress. The size of the board was limited to 80x100 mm as this is the maximum allowable board size in the free version of eagle. This proved challenging in the creation of the PCB.

All the components were placed on the board and it was auto-routed to give an indication of placement pattern. The result yielded 84.2 % routed (see Figure 12), which was not high and indicated poor placement. It was decided to place all the components and route manually.

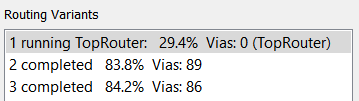


Figure 12 - Auto Routed Test of Placement

All of the components were successfully placed but the SDA processor which had 32 pins and proved difficult routing which originally could not be routed as indicated by the three air wires in Figure 13.

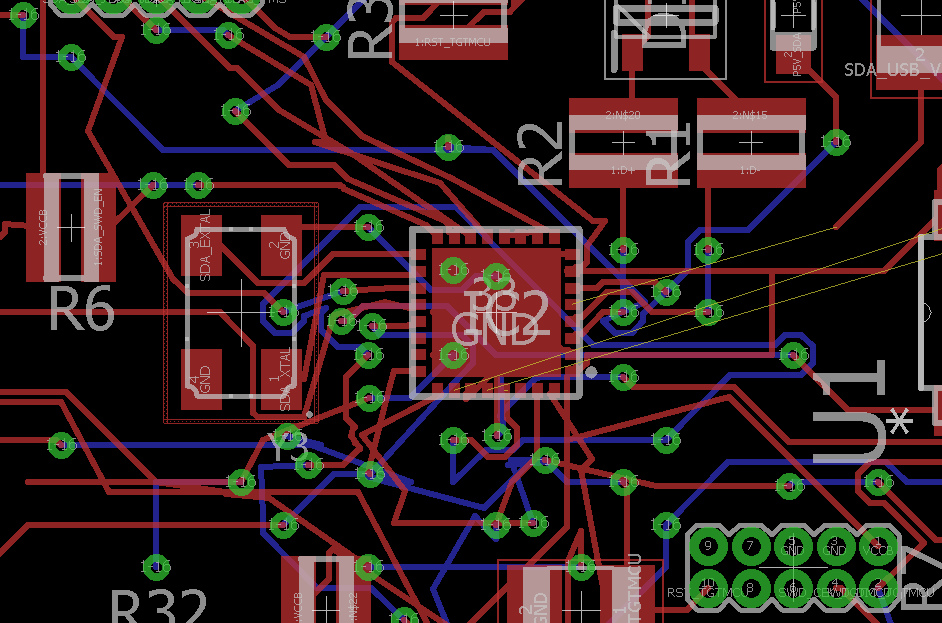


Figure 13 - SDA Processor with Poor Routing

Routing manually allowed for the optimal placement of the components, this included:

1. Placing all the CT circuits in their individual circuits off to the side
2. Increasing the free space around each processor for routing
3. Allow room between the FRDM headers for routing around the outside
4. Placing the power supply together in a corner
5. Relocating the USB outside of the FRDM Headers

The finished PCB is shown in Figure 14 and the areas of noted above are shown with their corresponding letters.

B

E

B

A

D

C

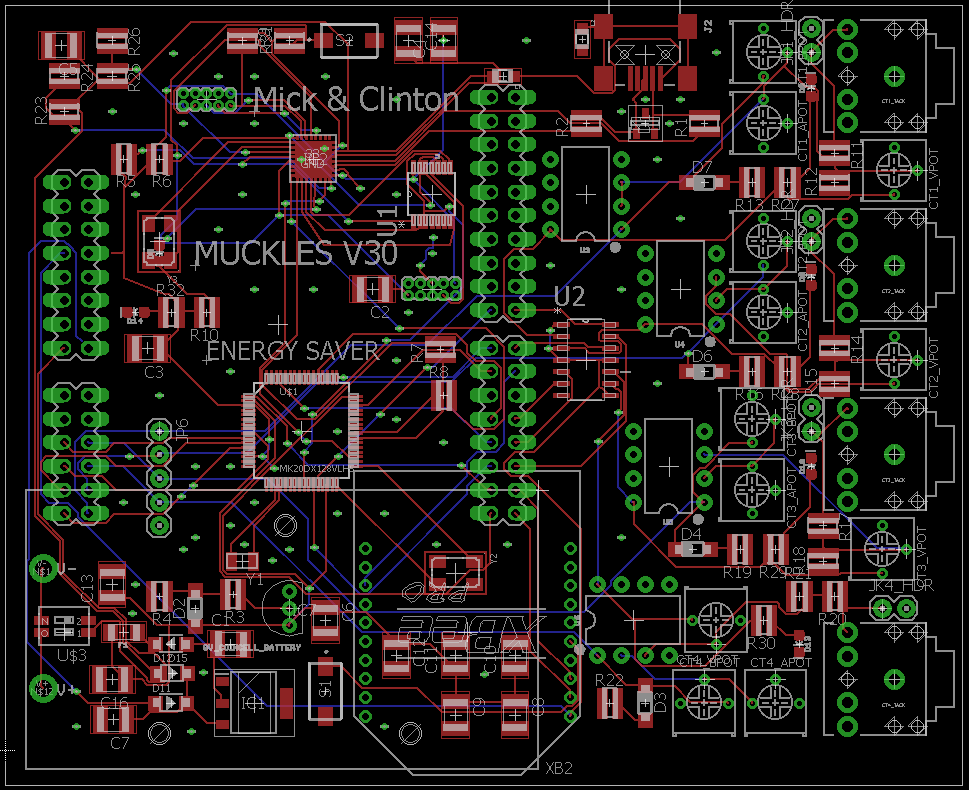


Figure 14 - Finished PCB

# SOFTWARE DESIGN

## Kinetis Code

### Analog to Digital Converter

The Kinetis code was developed by adding on each task successively. The first step was to setup the project and get it receiving data from the channels. The pins for the Analog to Digital Converter (ADC) channels were setup when designing the schematic. The advantage of using FRDM-K20D50M processor was the familiarity with the syntax of C and the onboard ADC.

### Timing

Figure 15 shows the operation of the circuit with respect to timing and defines a phase, cycle and period (note the units are seconds).

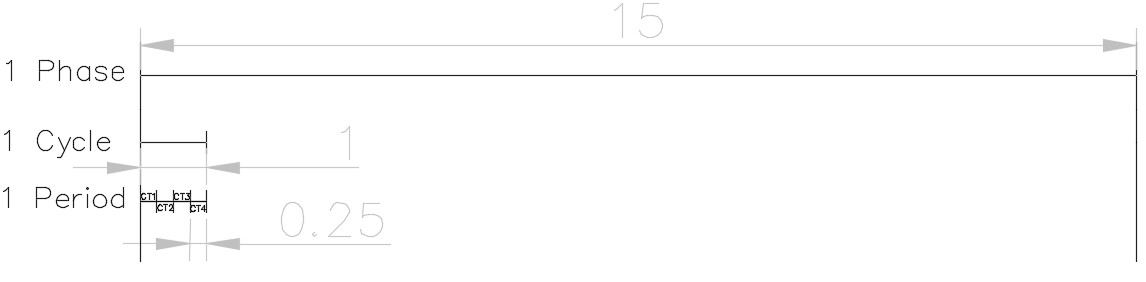


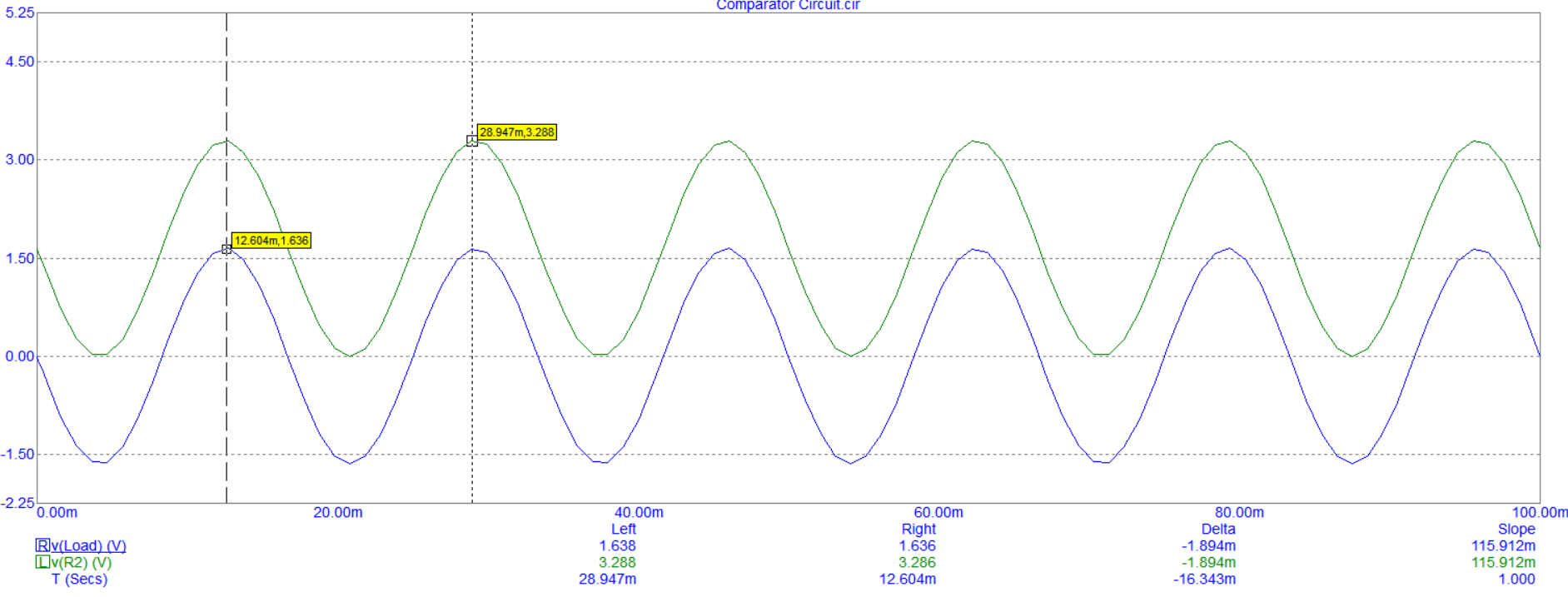
Figure 15 - Definition of Timing in Circuit (seconds)

Figure 15 shows there were three loops timing sessions occurring; a phase, a cycle, and a period. The program was sent into a wait for interrupt phase on startup while enabling Timer 1 which was a 15 second timer that timed a phase. This would interrupt and disable itself starting Timer 2. Timer 2 would interrupt itself every 1ms and on each interrupt a piece of data would be captured. After 250 of these data acquisitions a switch statement would turn on another channel until all channels were had data recorded. The switch statement would turn on a single channel at a time. A minimum value was set up that if a channel did not read above a (xx) minimum value then it was disabled for 40 phases which was 10 minutes.

### Data Percentage

The values to for each channel were sent to a putty to verify when testing and display the state of each channel. This was important during the calibration phase and a clipping range was introduced. This clipping function occurred if the input ranges of the ADC were above 3.300 V which translated to 16-bit number of 65,535- 0, while a buffer of 100 was placed on each side shifting it to 65,435-100. It was decided that it was easier to map this to a percentage which could be calibrated to the type of sensor (10A, 20A or 25A) during setup. This removed the necessary gain adjustment.

### Data Calculation

The data for a period was summed for each point of the 250.

The equations for RMS here ?

### ZigBee Transmission (MICK)

The transmission of the data over the Xbee module will be done by placing the data into array and using the Asynchroserial function to send one character at a time. The code shows the current sensor values placed into the buffer named message with the *strlen* function assigning the message size which is important before the API structures can be constructed.

**static** **char** message [100];

**snprintf**(message,100,"%f,%f,%f,%f\n",CT\_Current[0],CT\_Current[1],CT\_Current[2],CT\_Current[3]);

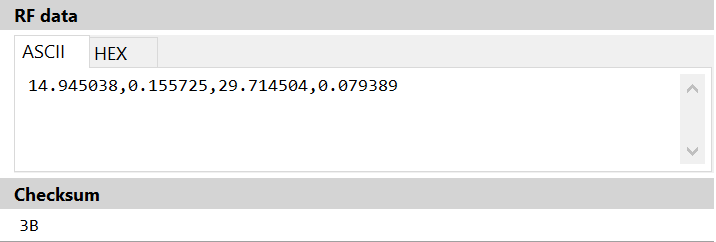
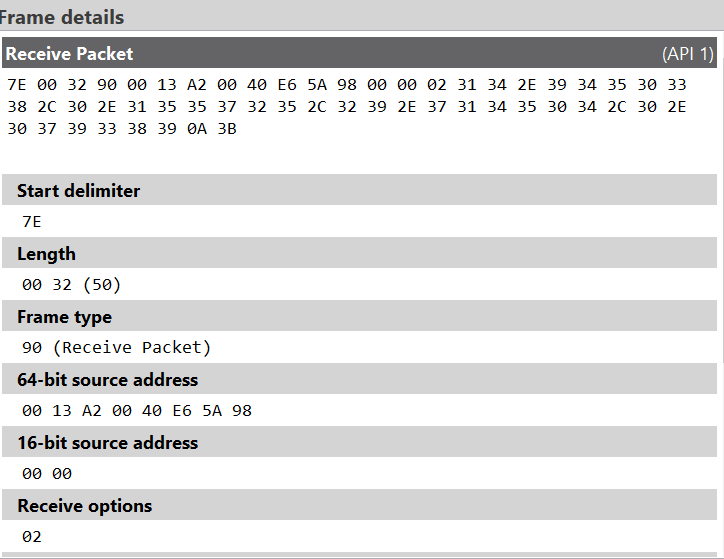
**int** message\_size = **strlen**(message);

Xbee’s provide a mode called Application Programming Interface (API) which provides uses with a structured interface where data is communicated the serial interface in orgnised packets in a determined order. Data transmitted in the form of API packets or data frames have a very well define structure and understanding this structure is crucial to derive data from the frame.

An API frame has the following structure

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Start Delimiter | Length | | Frame Data | | | | | | | | Checksum |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | … | n | n+1 |
| 0x7E | MSB | LSB | API-specific structure | | | | | | | | Single byte |

The specific structures can be found by connecting the Xbee to the XCTU program whilst the specific data is being sent. The picture below shows the API structures for the CT Monitoring Assignment.



This information is used to write the code in kinetis to construct the Xbee API packet as shown below.

// Construct the Xbee API frame packet

byte packet [128];

packet[0] = 0x7E; // Start delimiter

packet[1] = 0x00; packet[2] = (byte)(message\_size + 14); // Frame length

packet[3] = 0x10; // Frame type: Transmit request

packet[4] = 0x01; // Frame ID

packet[5] = 0x00; packet[6] = 0x00;

packet[7] = 0x00; packet[8] = 0x00;

packet[9] = 0x00; packet[10] = 0x00;

packet[11] = 0xFF; packet[12] = 0xFF; // 64 bit destination address: Broadcast

packet[13] = 0xFF; packet[14] = 0xFE; // 16 bit destination address: Broadcast

packet[15] = 0x00; // Broadcast radius

packet[16] = 0x00; // Options: None

The message can then be placed into the frame packet once the API structures are established.

// Place message into frame packet

**for** (**int** i = 0; i < message\_size; i++) {

packet[17 + i] = (byte)message[i];

}

The Checksum is the last byte of the frame and helps test the data integrity. To calculate the checksum of an API frame all the bytes of the packet excluding the start delimiter and length are added. The lowest 8 bits only are kept from this result and this quantity is subtracted from 0xFF. If the checksum is incorrect frames sent through the serial interface will never be processed. The Checksum calculation is shown below.

// Xbee API checksum calculation

uint8 checksum = 0xFF;

**for** (**int** i = 3; i < 17 + message\_size; i++) {

checksum -= (uint8)packet[i];

}

packet[17 + message\_size] = checksum;

The data can be sent using the asynchroserial fuction:

// Transmit one byte at a time

**for**(**int** i = 0; i < **sizeof**(packet); i++) {

**while**(AS1\_SendChar((byte)packet[i]) != ERR\_OK) {}

## C++ Code

### ZigBee Data Acquisition (MICK)

The receiving of the data packets on the raspberry pi required the use of unions and structures of the C++ programing language. A union in C programming is a user defined variable which may hold members of different sizes and type which all members share the same memory location. A structure is a convenient tool for handling a group of logically related data items. Structure help to organize complex data is a more meaningful way. The follow code was implemented to handle the Xbee protocol.

static union {

char buf [RXBUF\_LENGTH];

struct \_\_attribute\_\_((packed)) {

uint8\_t start\_delimiter;

uint16\_t length;

uint8\_t frame\_type;

uint64\_t source\_address\_64;

uint16\_t source\_address\_16;

uint8\_t receive\_options;

char rf\_data[]; // up until the end of the union.

// There is a checksum field immediately after the end of rf\_data.

} packet;

} rxbuf;

The incoming bytes are placed into rxbuf.buf[]. The struct rxbuf.packet allows for named access to particular fields within the binary protocol. The packet receive function was written with several if statements that reads from the serial port into rxbuf until a complete packet has been received. It returns the length of the data payload or -1 if receive failed. Before the loop of the function was run the buffer was zeroed and an index was initiated to 0.

memset(&rxbuf, 0, sizeof(rxbuf));

rxbuf\_idx = 0;

The API frame structure is known, therefore within code there are checks, as we are expecting the start delimiter, the following was written:

*if ((rxbuf\_idx == 0) && (c != 0x7E)) {*

this *checks* the data to see if the first byte is the expected start delimiter. If this does not occur we are not synchronised with the Xbee and we discard bytes by restarting this loop body until we see a start of frame delimiter. If the start delimiter is received then the characters are saved into the buffer. Once rxbuf\_idx is 3, this acknowledges that we have received the length of the packet. The received packet length is in big endian format. Endianness refers to the sequential order in which bytes are arranged into larger numerical values, when stored in computer memory, or when transmitted over digital links.

*rxbuf.packet.length = be16toh(rxbuf.packet.length);*

When rxbuf\_idx is greater or equal to 4 this indicates that the number of bytes received is rxbuf\_idx, this is due to 4 bytes not being counted in length which are the start delimiter, the 2 bytes of length and checksum.

} else if (rxbuf\_idx >= 4) {

The complete packet is received when we have the length plus 4 bytes.

if (rxbuf\_idx >= rxbuf.packet.length+4) {

The function performs a number of checks including the check sum calculation, to confirm it is indeed the required data, the function returns the data by subtracting 12 off the length as there are 12 bytes of header included in packet.length.

return rxbuf.packet.length - 12;

### WiPi Upload (MICK)

To connect the Pi to internet the Wi-Pi module needed to be configured to the network that we the device was operating through. The network configurations were modified in the command line editor to set up a wi-fi connection on the JCU network. transmitting the logged data is by sending specially crafted HTTP requests. A simple HTTP library is called “libcurl” is installed on the Pi. Libcurl is a free and easy-to-use client-side URL transfer library, it is a computer software project providing a library and command-line tool for transferring data using various protocols.

// Initialise the HTTP library

CURL \*curl = curl\_easy\_init();

if (!curl) {

printf("Failed to initialise the curl library\n");

return 1;

}

curl\_easy\_setopt(curl, CURLOPT\_WRITEFUNCTION, http\_callback);

### ThingSpeak Display (MICK)

The data stream was created by accessing the Thingspeak website, by signing up for a account, and creating a new channel at *thingspeak.com/channels.* Four independent data fields were specified as this corresponded to the four sensors to be read by the board. Once the channels are created the writing API key was found and copied into the code. The writing key is required to upload data to the channel, it provides a URL to use in the code.

// Scan the received data and construct url

sscanf(rxbuf.packet.rf\_data, "%f,%f,%f,%f\n",&CT\_Current0,&CT\_Current1,&CT\_Current2,&CT\_Current3);

snprintf(url,500,"https://api.thingspeak.com/update?api\_key=JYCQY04Q24PQWZPR&field1=%f&field2=%f&field3=%f&field4=%f",CT\_Current0,CT\_Current1,CT\_Current2,CT\_Current3);

printf("%s\n", url);

# TESTING & DISCUSSION

## Final Product

Table for calibration!!

## Game-Day Performance

The entire circuit was then built onto.

# LIMITATIONS & RECOMMENDATIONS

## Limitations

## Recommendations

Maybe zigbee switched on with the mux?

Use RTOS?

Better implementation of interrupts?

# CONCLUSION

Project #1: non-invasive current metering

Using CT (current transformer) sensors, monitor the AC current in external circuits. CT sensors will be supplied that produce an AC voltage in proportion to the current being measured. You will sense this voltage and thereby calculate the current that is flowing.

Your system must be capable of simultaneously monitoring at least 4 circuits per microcontroller.

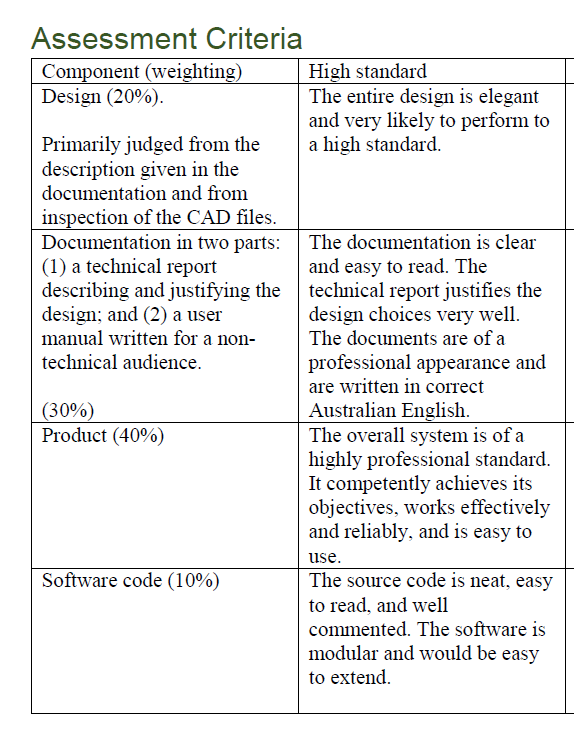
You must implement **at least one** of the following features:

• A sensor node that operates without access to mains electricity, e.g. using built-in batteries and/or scavenging power. If your device is battery powered then you do not need to implement wireless communication.

• Wireless communication between sensor node(s) and a base station, with near to real-time data upload to the Internet. Your base station will be a Raspberry Pi.

**Hints:**

• This project favours electronics and circuit design. You will need to implement some analog circuitry to sample the AC voltage, e.g. by DC biasing into the midpoint of the ADC range.



# REFERENCES

[1] E. Hub. (2017, 17/10*). Current Transformer*. Available: <http://www.electronicshub.org/current-transformer/>

LIST OF APPENDICES

[APPENDIX A – SCHEMATIC 1](#_Toc496114080)

[APPENDIX B – PCB LAYOUT 2](#_Toc496114081)

[APPENDIX C – BILL OF MATERIALS 3](#_Toc496114082)

[APPENDIX C – CODE 4](#_Toc496114083)

APPENDIX A – SCHEMATIC

APPENDIX B – PCB LAYOUT

APPENDIX C – BILL OF MATERIALS

APPENDIX D – CODE

APPENDIX E – FINAL PRODUCT

