D5: Team J Smart Meter Design Final Report

Yai Sagolsem (ys6g21), Mathyus Marshall-Panayiotou (mmp1u19), Islombek Karamatov(ik5g21), AoYang Leng (al7g20), Yiyang Hu (yh10u21), Gagan Vithanala (gv4g21)

University Of Southampton, Electronics and Computer Science

Contents

Fig. 1. Our final smart meter design in operation. Load switch and call indicators visible, alongside the two PV and Wind level power bars. Beneath is the large total power readout

1. Introduction and Project Vision

Our team began the D5 Smart Meter Project with good enthusiasm and a drive to accomplish more than what was expected. This report will document the design that was produced and the final outcome and reality of that initial enthusiasm as it was tested against the many unfortunate obstacles that presented themselves.

2. Team Roles and Coordination

Gagan Vithanala (gv4g21): Initially assigning themselves to working on the housing, Gagan helped to design and validate the design that was eventually produced, by producing accurate CAD models of the various connectors. After the housing was completed, Gagan helped to design some of the LCD iconography.

Yiyang Hu (yh10u21): Yiyang was initially the main lead for housing design, and worked with Gagan to produce the designs for the first review. Afterwards, designs were iterated with Gagan under the guidance of the project lead to produce the final housing design. Once the housing was finalised, Yiyang began to draft the initial pixel dimensions of the LCD graphics.

AoYang Leng (al7g20): Aoyang worked with the project lead to develop backup PSU as well as the initial interface circuitry. Unfortunately some time was lost during the project as Aoyang fell ill for a week.

Islombek Karamatov(ik5g21): Islombek worked with Mathyus on the algorithm and software, and also wrote part of the firmware (io.c and mainsreq.c). Unfortunately during the final weeks Islombek was unable to work on the project extensively for 6 days as he had to travel to Manchester as his Visa was almost invalidated but he still helped with the algorithm development and testing during that period.

Mathyus Marshall-Panayiotou (mmp1u19): Mathyus was the main software lead and worked extensively on the algorithm with Islombek, and worked with the project lead to integrate it with the firmware. Unfortunately during the last few weeks Mathyus also fell ill with Aoyang and was also unable to attend meetings.

Yai Sagolsem (ys6g21): With the most practical and software experience, Yai acted as the main project lead, overseeing all areas of the project, especially electrical, providing guidance and design improvements when necessary. Yai also wrote most of the firmware as well as the LCD driver. The project source can be found on Yai's GitHub at https://github.com/kaichodesu/sotond5

3. Power Supply

3.1. Initial AC to DC Regulation

Many designs for AC to DC regulation were considered, however, with the restrictions of size, cost, efficiency and safety, the main options presented were swithmode topology designs such as switched inductor or switched capacitor. Upon extensive simulations using PSpice with industry standard software such as Cadence's OrCAD, a great effort was made to produce a finished schematic and PCB that would be manufactured once and work flawlessly the first time.

3.1.1. Primary Side Regulated Flyback Topology

A PSR flyback converter based around the Texas Instruments UCC28730 flyback controller was chosen as it would provide full galvanic isolation while also using a secondary side wake-up monitor to allow for reduced low load parasitic power losses.

3.1.2. Optimising for Low-Load Efficiency

As a typical flyback converter is optimised for high load efficiency, our design had to be manually tuned with ZVS snubbers and careful simulation of MOSFET dv/dt to pinpoint component values that allowed us to avoid having to snub inductor voltage spikes where possible - greatly reducing the amount of energy lost while under low power conditions.

3.1.3. Capacitor Discharge Requirements

Unlike most AC to DC designs, we were able to greatly reduce our required input capacitance by many orders of magnitude - only using an input capacitance of 300 nanofarads. This allowed us to easily satisfy the safety capacitor discharge requirements, our voltage dropping below 30V upon mains disconnect almost instantaneously.

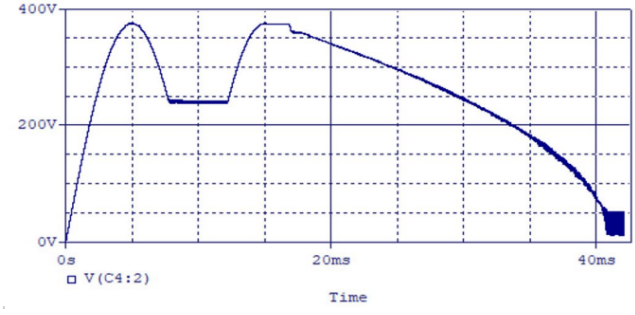

Fig. 2. Full discharge in less than 50 milliseconds.

3.1.4. Schematics

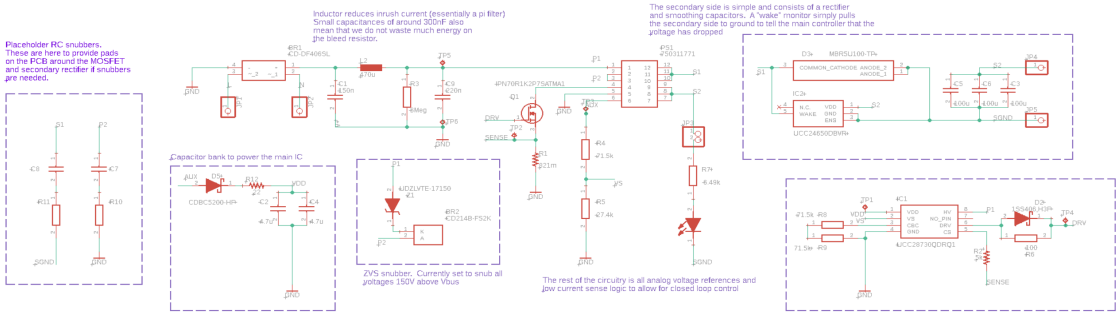

Fig. 3. The Final PSU Schematic.

4. Interface Circuitry

4.1. Introduction of Interface Circuitry

In order to connect the Test Bed to the smart Meter, an interface circuit system must be built, which consists of 8 small circuits in total. They are all signal processing problems about voltage transform. Four significant sections called Analogue input, Analogue output, Digital input, and Digital output. All input sections need to convert voltage from Smart Meter to the Test Bed and the output sections from the Testbed to Smart Meter.

4.2. Analogue Input

The main capacity in the analogue input section asks us to give the Test Bed a DC 0 10V voltage. Figure 1 shows that R1 and C1 convert the pulse to DC voltage, and the non-inverting Operation-Amplifier output the 10V DC voltage. PSU provides 15V to Op-Amp.

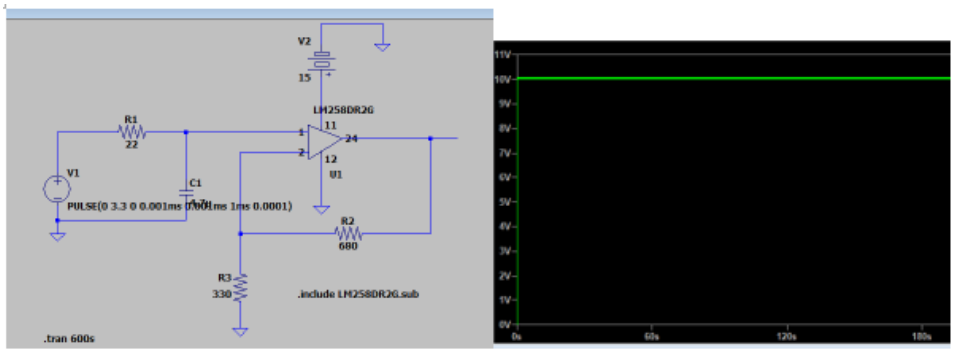

Figure 1: Main Capacity

4.3. Analogue Output

As part of the Analogue output, a busbar voltage should have a stable output of 3.3V from AC -4V to 4V, as seen in Figure 2. Using one inverting Op-Amp and one voltage follower to get the correct result. PSU provides 15V to Operation- Amplifier. One diode to convert AC to DC. One capacitor makes output voltage more stable. Furthermore, as busbar current circuit was also made from AC - 10V 10V to 3.3V, figure 3. The circuit is similar to the busbar voltage. Moreover, Wind and PV capacity circuits from DC 0V 5V to 3.3V are similar to the Digital output section, which will be introduced later, shown in figure 4.

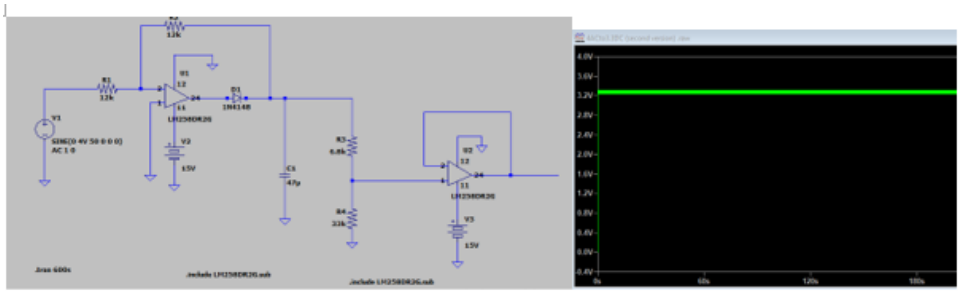

Figure 2: Busbar Voltage

4.4. Digital Input

Regarding the TTL logic voltage signal, the voltage needs to be transformed to a square signal 0 5V from the 3.3V Smart Meter for the Digital input section. In the TTL logic signal, 5V logic high means turn on, and 0V logic low means turn off. Our group decides directly output it from il Matto.

4.5. Digital Output

The last part is also about the TTL logic signal, which is from the Test Bed 0 5V to Smart Meter 3.3V. Hence, it has the same principle as the Wind and PV capacities in Analogue output. Similar circuit with a different source. Just a simple voltage follower.

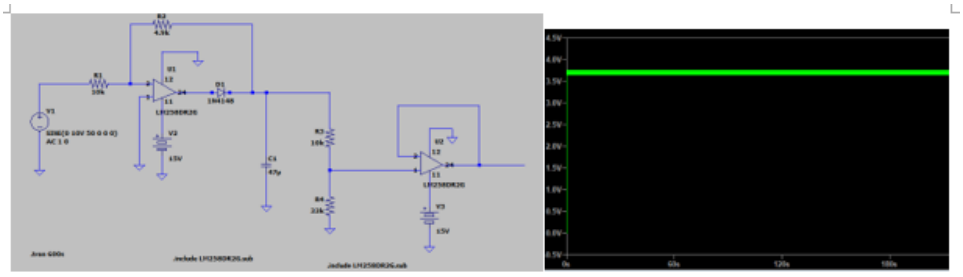

Figure 3: Busbar Current

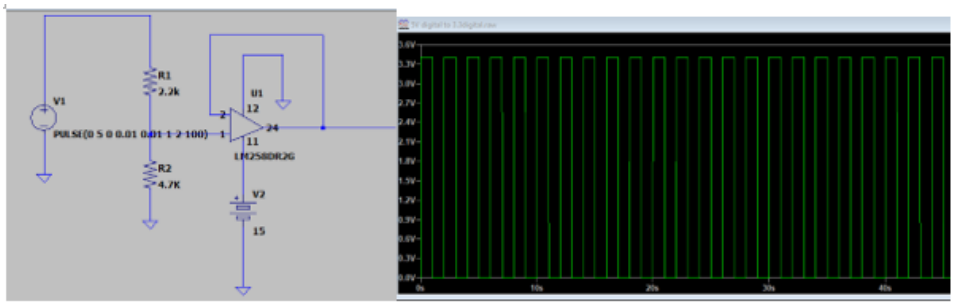

Figure 4: Digital output

5. Physical Design

5.1. Housing and CAD Modelling

5.1.1. Intro to Housing Design

The enclosure design needs to be convenient, waterproof and have clear viewer visibility of the readings on the LCD screen. It needs to implement these core principals along with the requirements stated in the specification. The box is limited to a 110x110x110mm dimension which should consist of a minimum of three parts. It needs to include a mounting lug so it can be hanged and five RJ45 ports along with a cable gland port. Lastly, the design should allow for easy access (within a minute) to the internal circuitry and needs to incorporate a segregated area for the PSU to prevent it from interfering with the other circuitry. The final design is influenced by outdoor socket enclosures, having a compact design and the use of O-rings and gaskets. The design consists of 4 parts: acrylic lid, main box, the electronics tray, and the PSU enclosure.

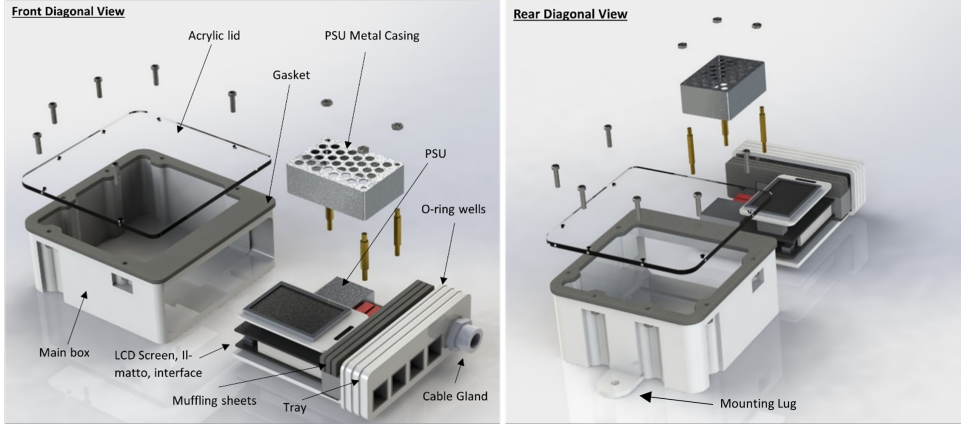

Fig. 4. 3D SolidWorks Exploded view of Housing

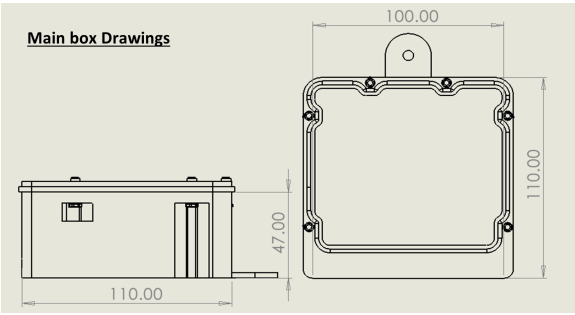

Fig. 5. Main box drawing

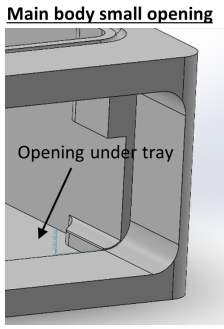

Fig. 6. Gap below tray

5.1.2. Acrylic Lid

The final box uses a laser cut acrylic lid on top. This allows the user to read information displayed on the LCD screen and plays its part in the box aesthetic, being able to see the circuitry inside. It aids with the tray mechanism too as everything is connected on the tray, rather than having the LCD attached to the main body surface, and therefore can be taken out from the box with ease.

5.1.3. Main Body

This is the main part of the housing that encloses everything. It is designed with screw tunnels which gives enough space for the nuts to be placed for the screws. The tunnels near the entrance tray gap are shorter than the rest to make sure it does not block the far left RJ45 when the tray is slid in. After printing the model, it was noticed that the tray and the main body were not interacting well when assembled and dissembled. This was due to the printing tolerance.This was fixed through sanding the inside well walls and the tray opening hole so the O-rings can fit in. The main body also has a small dip after the opening lip to ensure that the screw-heads which are underneath the tray don't prevent the tray from lying flat against the internal cavity.

5.1.4. Electronics Tray

We looked at various mechanisms from using screws to open the enclosure to a screwless tray mechanism. The tray design seen in figure 1 holds the PSU, Il-matto PCB stacked on Interface board (customised to fit in a compact space), and the LCD screen. Holes are drilled into the tray platform into which M3 screws and standoffs hold the PCBs in position. The tray can be ejected so the circuitry can be accessed. The decision of using no screws, to keep the tray in, allows for us to access the internal circuitry well under a minute. We ensured the tray , when inside the main box, stays and supports the weight with all components with the use of O-rings fitted in the three 1.5mm diameter groves. This means we can use the tension and frictional forces between the tray, O-ring, and the box wall to make sure that the tray stays in when closed. Spacing in between cutouts for the RJ45s and the cable gland was a concern. The cut-out positions were determined by building SolidWorks models of the RJ45 and cable gland.

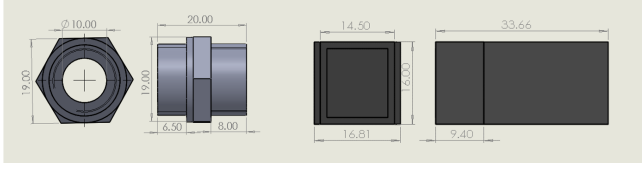

Fig. 7. RJ45 and Cable gland CAD Models

We were tight with a tolerance of 0.69mm. This resulted in the 3D printed model to just about fit all the RJ45s beside each other with no minimal gap.

5.1.5. Waterproofing

Gaskets and O-rings are mainly used for waterproofing. A gasket is used in between the acrylic lid and the main box. These three parts are fixed together tightly using M3 screws and nuts, ensuring minimal water

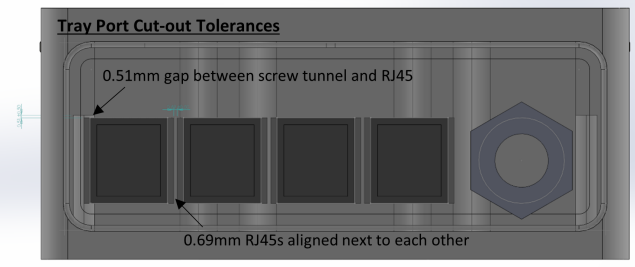

Fig. 8. Front view drawing of Tray

ingress within the gap. Three laser-cut gaskets are used as muffling sheets which are placed around the Ethernet ports and cable gland to stop any water leaking through the cut-outs. These design choices have proved themselves handy as we got a result of 0mm in the water test, proving the housing is highly waterproof.

5.2. PCB Layouts and Manufacture

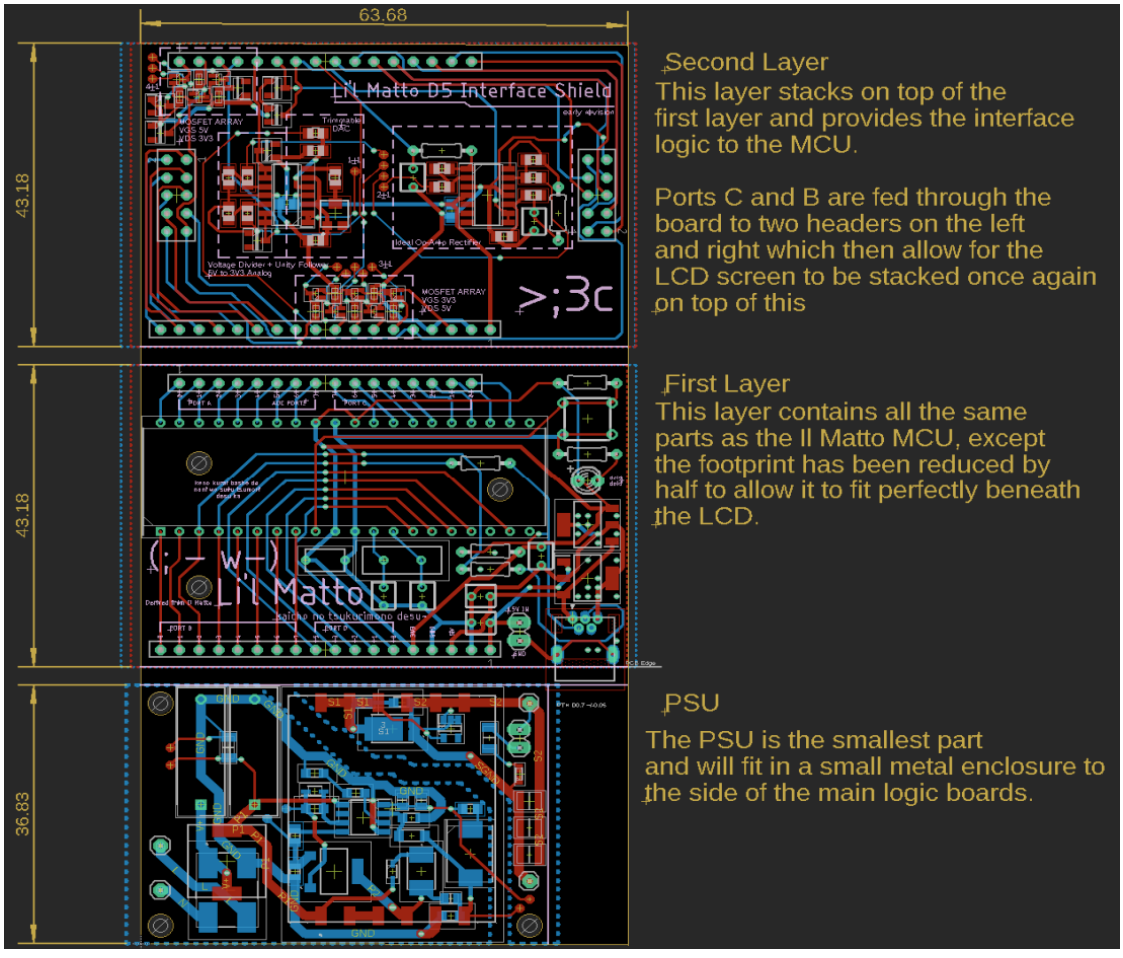

Fig. 9. The final triple board PCB design.

6. Firmware

In order to achieve the highest accuracy of data read possible, an advanced firmware had to be designed to run on our micro-controller to provide time-accurate data for the algorithm software to make reliable decisions. Our firmware was also designed to further limit power consumption as much as possible by using MCU sleep states, further contributing to our incredibly low quiescent power draw. To limit our BOM, we also opted to use the AtMega1284P as it was already available in the labs, but also featured double the program memory, with 128kB rather than the limiting 64kB that came standard with the Il Matto.

6.1. AC Phase Capturing and Clock Alignment

As our interface design did not use a capacitor to smooth the analog AC signal, we had to initialise a global tick that would send an interrupt at 50Hz to allow our ADC to sample the bus current and bus voltages at the peaks of their cycles. This was done by initialising one of our 8-bit timers on power up such that it would run at 50.08Hz and send an interrupt whenever its Output Compare Register overflowed.

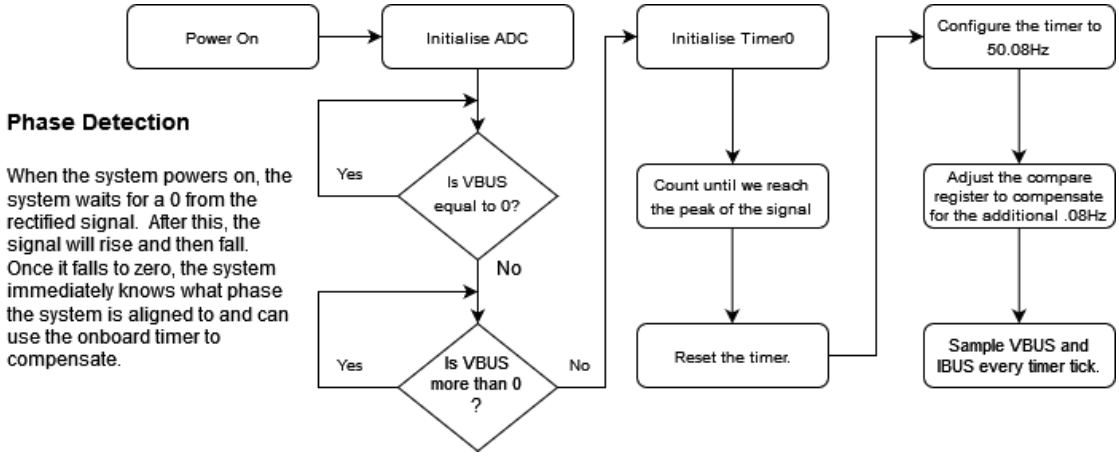

Fig. 10. A simplified flowchart of our ADC initialisation.

This system essentially acts as a zero-crossing-detector and also allows the system to execute the main software *after* the time-critical bus voltage and bus currents have been measured.

6.2. Noise Rejection

Due to the large amount of electrical appliances in the testing area, we discovered very quickly that it would be necessary to implement some filters in programming to mitigate the effects of these noise sources. Thankfully, the implementation of these filters was relatively simple as the only consideration that had to be taken was to ensure that the total thread execution time did not exceed 10 milliseconds as this would mean the system would overrun its internal 50Hz tick.

Thankfully, due to the relatively high sampling frequency of the system, it was found not necessary to implement these filters during normal operation, as taking an average of five consecutive readings was enough to nullify any random spikes.

However, due to the time-sensitivity of the phase detection code, we had to implement a for loop during each ADC event that would write the ADC results to a 20 value array. As we did not care for the *true* magnitude of the reading - since this was only to detect a zero-crossing - we simply had to make sure that any datum that was disproportionately large to the rest of the values in the array were ignored.

6.3. MCU Sleep States and Efficiency

By utilising the aforementioned 50Hz timer interrupt, not only do we ensure that our entire systems operates with up-to-date data, but we can also utilise the deterministic nature of this process to save power when the system is idle. Since we can observe how long instructions take to complete, we can then, when possible, put our micro-controller to sleep at the end of the thread, rather than have it constantly operating at full load, constantly sampling for an input that isn't going to change. Instead, we use the Timer0 Interrupt as

a signal to wake the MCU and compute *only when required*, reducing the power consumption of our MCU by up to 80%.

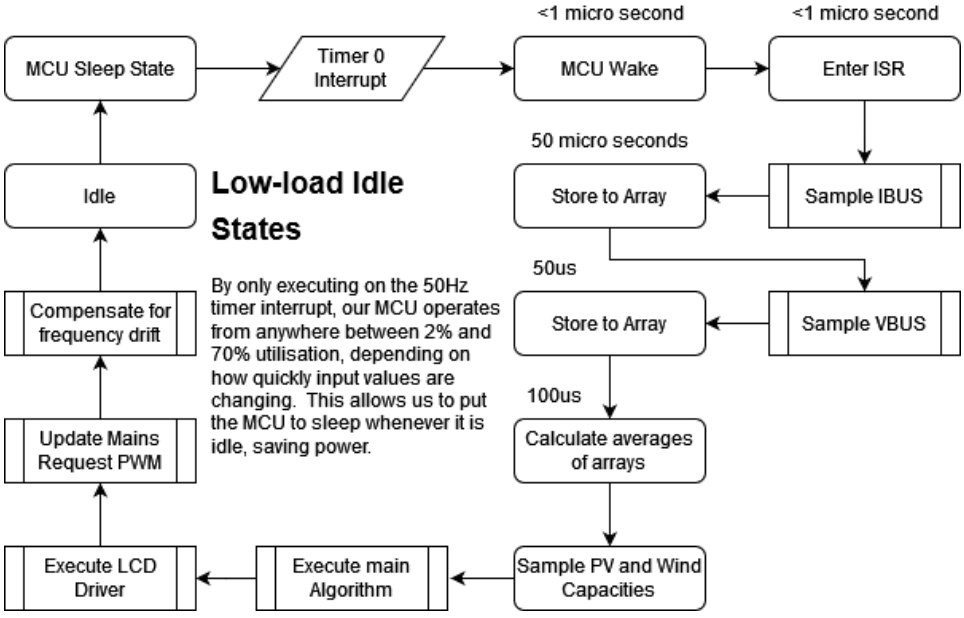

Fig. 11. A simplified flowchart of our Interrupt Service Routine.

6.4. Display Driver

In order to display appealing graphics to the end user, we had to design an LCD driver that would not only allow us to directly address the LCD driver chip's data registers, but also do so in a way that did not compromise the timing cycles of the Phase-Synced interrupts. We found that the display instructions took a few hundred milliseconds to execute if we were to update the entire screen, which would prevent our algorithm from running effectively as it would delay the reading of the ADC and thus provide erroneous data for the calculations. Thus, we decided to implement an LCD driver that dynamically updates parts of the screen only when necessary.

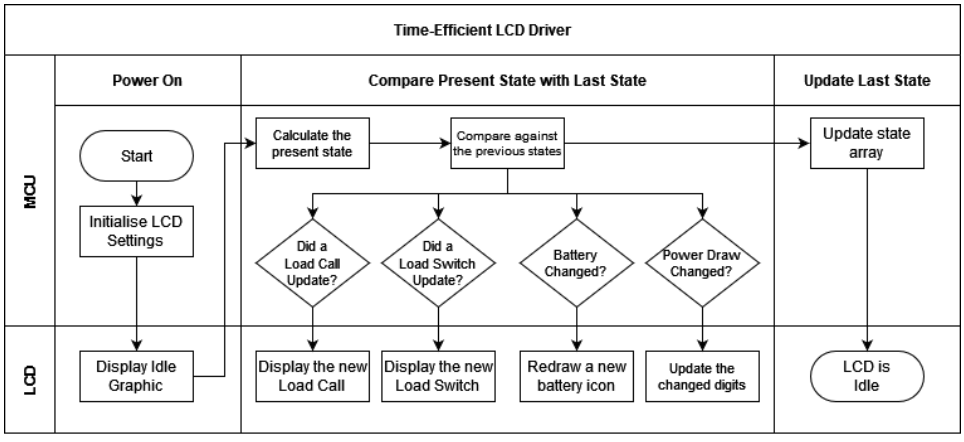

Fig. 12. A simplified flowchart of our LCD Driver.

This allowed us to update parts of our screen at 50Hz, after the phase-aligned ADC measurements had been taken - giving the user a smooth visual indicator of our smart meter's actions.

6.4.1. Memory Addressing and Memory Management

One hurdle we had to overcome was the difficulty of programming for an 8 bit micro-controller with more than 64kB due to the memory addressing issues present - standard 16 bit values cannot address the higher memory ranges. This isn't usually an issue for the end user, however, it was for us since we were storing

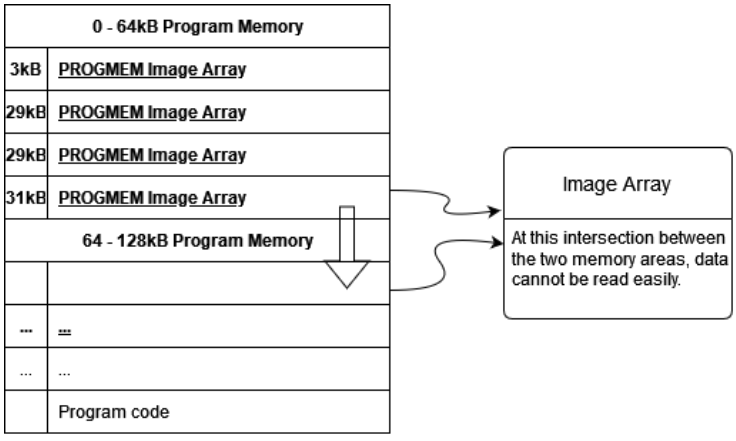

Fig. 13. Highlighting the issues when an image array overlaps the boundary between 64kB and 128kB.

image bitmap data directly in the program memory, rather than loading it into SRAM as is typically done in normal code. This issue was eventually overcome by manually searching for the assembly instructions compatible with the 1284P family of micro-controllers and defining macro functions around them to produce a set of functions we could use to interact with data outside the typical 16 bit address range.

6.4.2. Texture mapping

Wanting to make our user interface as appealing as possible, we invested into creating our own font set to display larger, more legible numbers on the screen. Naturally, this came with a problem that had to be solved by looking at the methods used by developers in the past when once restricted by similar low memory, low processing power environments.

Fig. 14. A texture containing all the digits used for our power readout.

Similar to the 8 bit computing era of digital entertainment, we also use texture maps to display our numbers. Each number is addressed by calling it from its offset, allowing us to then draw the pixels after the offset to display the desired digit.

7. Software

7.1. Algorithm

The algorithm runs off the basis that each load call can be represented as a binary number. Three loads represents 8 possible combinations of loads being either on or off (000-111). Firstly, we check which combination of loads are on and decide what to do accordingly.

7.1.1. Supply and Demand

Loads are prioritised according to their number, ie Load 1 takes highest priority and Load 3 is lowest priority. If a combination of loads are on and the energy requirements are not met from renewable energy sources, we begin to request from the mains. Bus voltage will drop when renewable sources are not providing sufficient power, so we increase our request from the mains supply iteratively until the bus voltage is at its normal specified value. The battery discharges to provide extra current if the difference between the max supply and loads that are currently switched on, are less than the value required to charge the battery.

7.1.2. Backup Generator

Initially we decided to choose whether to charge/discharge the battery outside of the branches, according to bus current. This worked for load values simulated before the final profile was released, however the algorithm became stuck in a loop, constantly charging then discharging. This happened due to the bus current decreasing if we choose to discharge the battery, therefore the algorithm would see enough space to charge the battery and repeat.

We decided to place the decision to charge the battery based on constant load values within each branch. If the condition above is met (max supply minus Loads switched on is less than Battery charge current), we discharge the battery, otherwise we charge it.

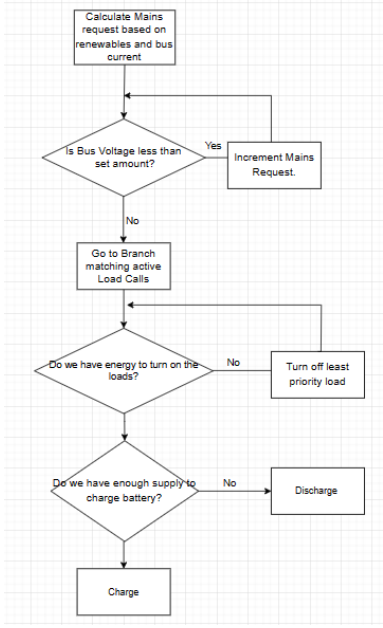

Fig. 15. Simple flowchart showing the basic functions of algorithm.

7.1.3. Equations and conditions

The general equation used to calculate how much to request from mains is as follows:

$$Mainsreq = \frac{10}{MainsMax} * BusI - (PV + Wind)$$

We cannot request a negative amount from mains or above 10, so two simple if statements were used to

limit this:

$$if(MainsReq > 10)MainsReq = 10$$

$$if(MainsReq < 0)MainsReq = 0$$

Deciding whether to charge or discharge the battery is done using the following general equation and is dependant on what branch of the algorithm we are currently in:

$$if((MainsMAX + PV + Wind) - sumOfLoads < BatteryChargeI)$$

$$discharge$$

$$else$$

$$charge$$

fig.16 shows how we choose to turn loads on or off and gives a good description of load call dependant branching.

8. Conclusion

Overall this project has been a mixed experience with lots of areas that could have been improved upon. While there was great opportunity for the development of a truly outstanding design, unfortunately due to a lack of group morale and motivation mixed with poorly timed absences, the workload left for those who remained was simply too great.

Time was lost chasing red herrings a result of erroneous power readings during the second review and the project lead failed to audit group progress as aggressively as they should have to ensure that the reported progress was accurate to the current state of the project.

While it could be said that the goals set out were unrealistic for a short project, it is also fair to suggest that, seeing as the final design produced was only a couple days from perfection, had there been less absences, the goals set were wholly achievable.

However, despite having failed to produce an adequately performant design, the level of complexity and integration achieved in the time frame is at least something that the group can be proud of.

9. Appendices

9.1. Meetings and Minutes

Meeting Date	Location	Total Duration	Participants	Summary
30/01/2023	B16	120 minutes	Yai, Mathyus, Yiyang, Islombek, Gagan, Leng	Overview of specification, discussion of initial ideas
31/01/2023	B16	120 minutes	Yai, Mathyus, Yiyang, Islombek, Leng	Discussion of planning, setup of Gantt charts, potential circuits, and initial ideas for housing designs.
02/02/2023	B16	90 minutes	Yai, Islombek	Flyback PSU parameter finalisation and housing integration considerations; Renewable energy and battery usage optimisation ideas & DAC circuit considerations
03/02/2023	Online (Discord)	120 minutes	Yai, Islombek	Provisional circuit schematics; DAC circuit prototype simulation and testing
04/02/2023	B16 + Online	Entire Day	Yai, Yiyang, Gagan,Leng	PSU Simulation results, Housing designs and Interface designs.
07/02/2023	B16	120 minutes	Islombek, Mathyus	Display MCU interface mode discussion of ideas
08/02/2023	B16	180 minutes	Yai, Mathyus, Islombek, Gagan, Leng	Housing Designs, Interface and presentation preparation; II-Matto display pin mappings & prototype display control program flowchart
09/02/2023	B16	120 minutes	Islombek, Mathyus	Initial UI design
10/02/2023	B16	180 minutes	Mathyus, Islombek	Initialisation of display & first design algorithm planning
11/02/2023	B16 + Online	Entire Day	Yai, Mathyus,Gagan	Presentation Preparation
13/02/2023	B16	180 minutes	Yai, Mathyus, Islombek,Gagan,Leng	Post-Presentation Team Evaluation; Algorithm tweaking
14/02/2023	B16 + Online	180 minutes	Yai, Islombek	PCB Initialisation; PIN mappings finalisation
15/02/2023	B16	Entire Day	Yai, Gagan	Housing and Physical design Finalisation
16/02/2023 - 24/02/2023	Zepler	Various	Yai, Mathyus, Islombek, Yiyang, Gagan, Leng	Housing Manufacture, 1st draft algorithm + labview testing, pin mappings
24/02/2023	B32 and Zepler	420 minutes	Yai, Mathyus, Islombek	PCB Assembly, ADC peak detection planning, PWM & DAC planning
26/02/2023	B16	180 minutes	Yai, Mathyus,	LCD geometry functions and Housing refinement
01/03/2023 - 06/03/2023	Zepler and Online	300 minutes	Yai, Mathyus, Islombek	PSU troubleshooting, ADC peak detection 2nd draft and final LCD interface plan, Algorithm efficiency improvements
08/03/2023	B16	300 minutes	Yai, Mathyus,Islombek	Firmware Code Design
14/03/2023	B16	180 minutes	Yai, Mathyus,Islombek	Algorithm and DAC final bug fixes & Final Profile

Fig. 16. A table of our group meetings.

9.2. Final Bill of Materials and Parts List

Arrived?	Part	Description	Supplier	exVAT	Total Cost
	UCC28730DR	PSR Flyback Controller	Mouser	1.33	1.596
	UCC24650DBVR	SS Wake-up Monitor	Mouser	0.53	0.636
	IPN70R1K2P7SATMA1	N-Channel Power MOSFET	Mouser	0.67	0.804
	750311771	Flyback Transformer	Mouser	11.15	13.38
	CD-DF406SL	Bridge Rectifier	Mouser	0.711	0.8532
	MBR5U100-TP	SS Rectifier	Mouser	0.571	0.6852
Checked For Stock	Header Pins	-	UoS	0	0
Checked For Stock	SMD Resistors	Various 125mW	UoS	0	0
Personal Stock	SMD Capacitors	Various 0805 35V MLCC	Personal Stock	0	0
	CD214B-FS2K	High VRRM Diode	Mouser	0.414	0.4968
	UDZLVTE-17150	150Vz Diode	Mouser	0.248	0.2976
	CD214BFS2K	Fast Recovery Rectifiers	Mouser	0.414	0.4968
	DIOM8059X290N	Schottky Diodes & Rectifiers 200V, 5A	Mouser	0.645	0.774
Personal Stock	470uH Power Inductor	Salvaged from old VRM module	Personal Stock	0	0
	R523I322050P0K	2x Safety Capacitor	Farnell	0.59	0.708
	SSM3K15AFS,LF	8x Power Mosfet	Farnell	0.77	0.924
	LP324D	2x LP324 Op-Amp	Farnell	0.34	0.408
	BZX84-C3V3,215	8x 3.3Vz Protection Zener	Farnell	0.48	0.576
Checked For Stock	II Matto Kit	All II Matto components	UoS	0	0
	PSU, Interface and MCU PCB	Free PCBs for first purchase. £7.90 shipping	ALLPCB	-	7.9
	NSVR351SDSA3T1G	230mv Vf Schottky Diode	Farnell	2.46	2.952
	MCP1253-33X50I/MS	1MHz Charge Pump controller	Farnell	1.54	1.848
	MAX5035BUPA+	125KHz Buck Converter IC	Farnell	1.6	1.92
				TOTAL	**37.2556**

Fig. 17. Bill of Materials

9.3. Team Gantt Chart

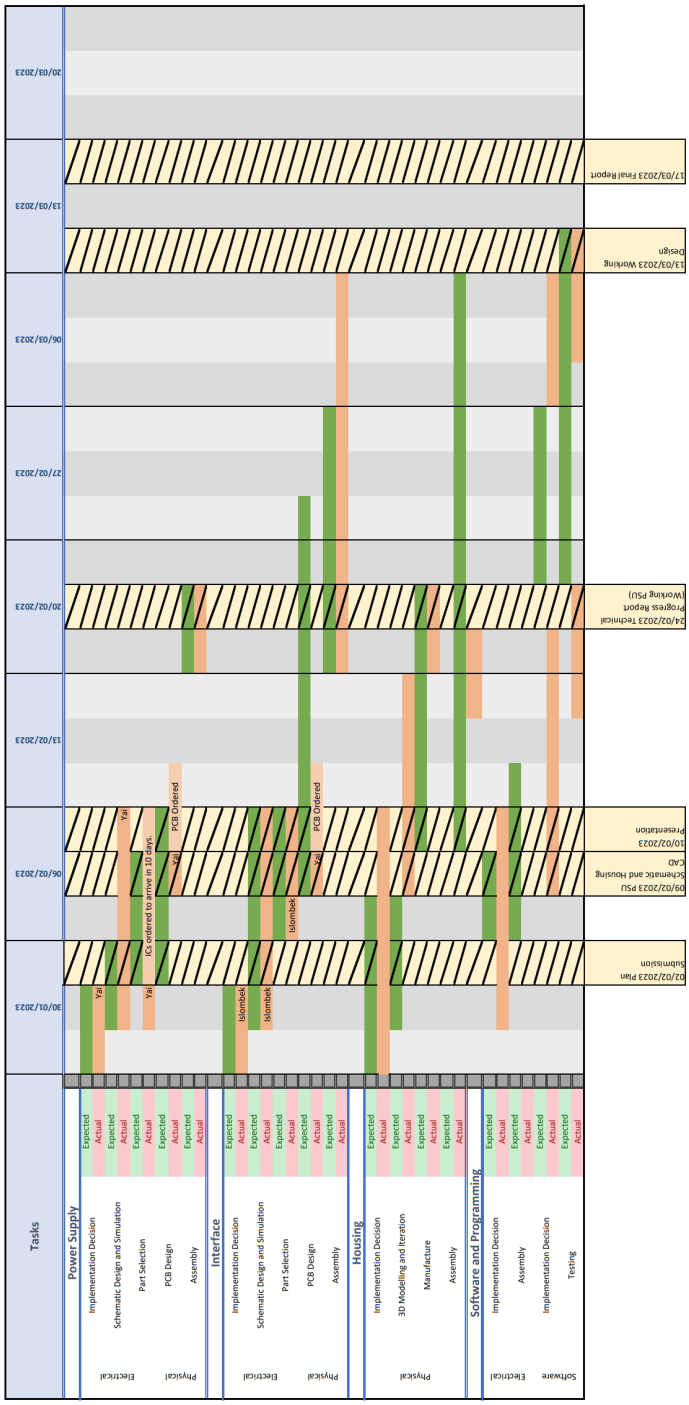

Fig. 18. Final Gantt Chart