

The Sustainable Innovation Fund: round 3
Project 92032

“Off-grid multi-energy reporting system and AI load controller”

Full Technical Report

Including System design and documentation, Prototype testing & AI system design

March 2021, revised Sept 2021

Natasha Wilson



**SMART
VILLAGES**
RESEARCH GROUP

Contents

2	Intro & Need	4
3	Technical Requirements.....	5
4	System Placement.....	6
5	Microcontroller	6
5.1	Arduino Nano	8
5.1.1	ESP8266 Wi-Fi module	8
5.2	ESP32.....	9
5.2.1	ESP32 ADC limitations and solutions	10
6	Sensing circuits.....	12
6.1	DC Voltage.....	12
6.1.1	Potential Divider	12
6.2	DC Current.....	17
6.2.1	DC Hall Effect Current Sensor	17
6.2.2	Current Shunt.....	22
6.3	AC Voltage.....	23
6.3.1	Voltage Transformer	23
6.3.2	Potential Divider	27
6.4	AC current	33
6.4.1	Split Core Current Transformer.....	33
6.5	AC software library	35
7	Data Logging.....	35
7.1	Local Storage – SD Card	35
7.1.1	Theory	35
7.1.2	Test Results	36
7.2	Writing data to the cloud.....	38
7.2.1	ThingSpeak.....	40
7.3	Keeping Time.....	42
7.3.1	NTPClient over Wifi.....	42
7.3.2	RTC module	42
7.3.3	Future improvements	43
8	Diverting DC Energy	44
9	“AI” switching logic	45
10	Full Wiring diagram	48
11	Prototyped product	50
11.1	Wiring.....	50

11.2	External connections.....	53
11.3	DC current connections	53
11.4	Solid State Relay.....	54
11.5	Enclosure.....	54
11.6	Final Prototype.....	54
12	Prototype Testing and Evaluation.....	57
13	Future Improvements Summary.....	58
13.1	Electronic hardware improvements:	59
13.1.1	Urgent- necessary for safety reasons, or to meet the initial technical requirements..	59
13.1.2	Highly desirable – to improve functionality.....	59
13.1.3	Non critical	59
13.2	Practical recommendations for future improvements:.....	60
13.2.1	Urgent- necessary for safety reasons, or to meet the initial technical requirements..	60
13.2.2	Highly desirable – to improve functionality.....	60
13.2.3	Non critical	60
14	Conclusion.....	61
15	Appendix	61
15.1	Resistor Sizing Calculations.....	61
15.1.1	1000V DC resistor calculations.....	61
15.1.2	50V DC resistor calculations.....	61
15.1.3	AC Potential Divider calculations	62
15.1.4	AC 100A Current Transformer Burden Resistor	62
15.2	BOM	64
15.3	Code	67
15.4	Interesting Similar Projects	70
15.4.1	Expandable 6 Channel ESP32 Energy Meter Main Board [15]	70
15.4.2	Home Energy Monitor (v2) [11]	70
15.4.3	PV Module Monitoring System Based on Low-Cost Solutions [16]	70
15.4.4	A Project to Design and Build a Controller to Route Surplus P.V. Energy into the Domestic Hot Water Supply [13]	70
16	References	70

1 Intro & Need

SVRG develops and supplies technology solutions and business models for crosscutting integrated rural development, particularly in sub-Saharan Africa. The strength and USP of SVRG's solutions is this broad integrated approach, that works with communities to develop a systems approach to simultaneously address their main needs and priorities, usually catalysed by access to energy, ICT, and enhanced agricultural productivity.

Most of our community development solutions have been powered by clean offgrid solar electricity. We have noticed a particular challenge in monitoring and controlling these systems, in that every equipment manufacturer tends to have their own bespoke monitoring and data collection system. These typically involve the equipment having an internet connection, and setting up an account via the equipment manufacturer's website.

The problem of this is fourfold:

- 1) It makes it more challenging for the local operators to access the data about their own systems
- 2) If we were just installing a standardised system everywhere we worked, this would be easy, and we would have just a single sign-in to access the data for all the components of every system we had installed – however, for a customer/community focussed approach like ours, where we design each system specially to meet particular needs, using whichever piece of equipment is the most appropriate, this might mean having 3 or 4 different accounts on different manufacturers' websites to collect the data on the systems. To date, we have installed equipment from 8 different manufacturers in the various communities in which we work
- 3) Some manufacturers charge for access to their data analysis and monitoring sites (i.e. you have to pay to get your own data)
- 4) It seems impossible to get data from some of the items of equipment – one of our frequently-used items, from a large and reputable manufacturer – does not seem to be able to just report data. We have been in conversation with their technical department for the past 6 months, and even they cannot make their own equipment just report data, rather than integrate into a large and expensive remote system-control application.

Furthermore, it is very difficult to apply sophisticated automated control decisions with much of this equipment. For example, solar panels often supply an excess of power, which the system cannot use. In circumstances like this, excess power could be flowed to an auxiliary load – our communities have told us a priority for this would be electric water heating. But there is no readily available device that has the functionality to switch this excess supply from solar panels without impacting the main power use, or to be able to control it in a nuanced and intelligent way.

The nearest equivalent (which doesn't perform any of the measurement or data recording, just the load diversion) is the dump or diversion load controllers that are used in sustainable power systems once batteries are full, to power an auxiliary load. But critically, these divert excess power from within the system (once a battery bank is full, for example) – we are suggesting diverting excess power before it reaches the system. A traditional diversion load controller in a solar system will divert the power produced by the solar charge controller, once the battery bank is full. We are suggesting diverting the excess power before it reaches the charge controller, which is otherwise inaccessible, since the charge controller simply will not draw that load from the solar panels.

For example:

- Solar panels (capable of producing 10kw – oversized in case of cloud)
- Charge controller (capable of drawing 5kw from panels and charging batteries)
- Inverter (capable of producing 5kw power from battery bank)

Take a sunny day, at noon, when the batteries are charged and the inverter is drawing/producing 2kw of power. A traditional diversion controller will divert the 3kw excess production of the charge controller, which cannot be used by the inverter or the batteries, to an auxiliary load. The system we are proposing will additionally divert the 5kw excess the panels are producing, but which the charge controller cannot physically draw, to the auxiliary load.

2 Technical Requirements

Table 1 lists the technical requirements as identified to maximise the usefulness of a monitoring system, installable in any of our solar installations, and giving a rapid and detailed dataset of system parameters at all the critical points, in an immediate and easy format both locally and remotely. It should also allow for diversion of excess power to a productive load such as heating hot water.

Table 1: Offgrid Energy Reporting System Technical Requirements

Category	Requirement	Notes
Sensing – All must work across multiple technologies	Sense 12 to 1000 VDC, Accuracy +-2V - Two sensors, one for up to 50V, one for up to 1000V	E.g. Higher voltage from a string of panels, lower voltage from battery bank
	Sense up to 300A DC, Accuracy +-2A -Two sensors, one for up to 50A, one for up to 300A	E.g. lower current from a string of panels, higher current from a battery bank.
	Sense 100-500V AC, Single and 3 phase (star and delta) Accuracy +-2V	E.g. following inverter or VFD
	Sense up to 100A AC Single and 3 phase (star and delta) Accuracy +-2A	E.g. following inverter or VFD
Data logging	Take accurate time-stamp for each measurement	For later analysis if desired
	Sample measurements at frequencies from 1s to 1hr	Most likely to be used in 1-5 minute range
	Write measurements to local storage device	E.g. SD card
	Communicate over internet to write data to cloud	
Power diversion	Programmable to apply a 5V switching signal to a solid-state relay to operate a dump load	E.g. for diverting excess solar energy into heating water
Cost	Components cost less than £100 total	
Installation	Can run from a 5V DC supply	
	External connectors, for easy installation in any location without soldering.	E.g. MC4 connectors for interfacing with solar equipment. Screw-in terminal blocks for AC measurements?

Safety	All electronics enclosed within an insulated or grounded box, so there is no risk of electrocution if outside is touched during operation	
	Contain appropriate fuses/diodes to prevent overloading microcontroller or other electronic components	

3 System Placement

Figure 1 shows the potential sensor placement in a generic solar system. Note that the placement of sensors on wires means that it can be installed independent of the charge controller and inverter equipment used, by intercepting the cables between equipment.

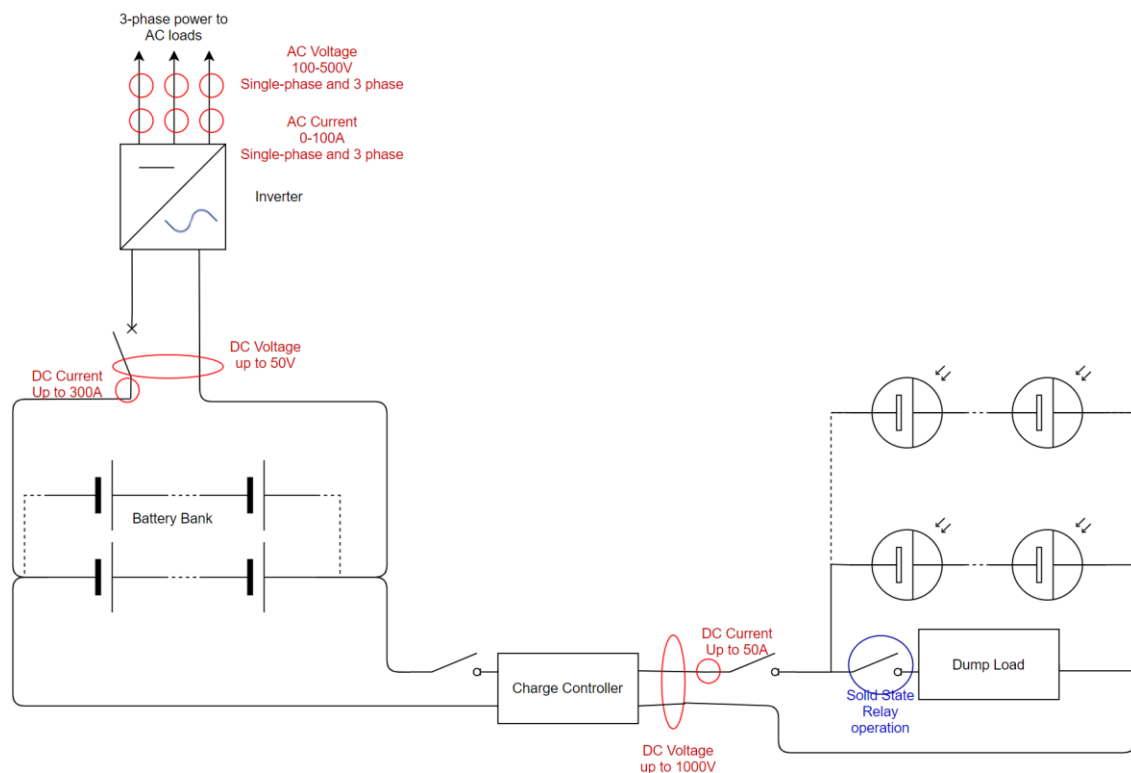


Figure 1: Wiring diagram showing sensor and dump load placement in generic solar system. Battery bank and solar panel strings and arrays are shown with dotted lines to indicate potential for expansion.

4 Microcontroller

The microcontroller is the main interface behind the sensing device, taking information from all the sensors, saving and uploading the data, and processing it as necessary to trigger dump loads. From the technical requirements, we can draw the following core functionalities as required by the Microcontroller chosen:

- Read in up to 10 sensors, through inbuilt analogue input or from external ADC chip (interface SPI or I2C)
 - 2 DC voltage
 - 2 DC current
 - 3 AC voltage

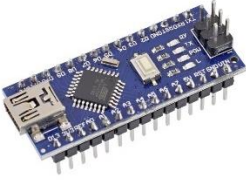


- 3 AC current
- Record data onto microSD card (Uses SPI connection)
- Connect to Wi-Fi and write data to cloud
- Keep track of time even if power is lost
 - Use an RTC module (I2C connection) with a small battery
- Control a 5V solid state relay to turn on/off a dump load

A number of microcontrollers were researched, namely:

1. The Arduino Nano: a cheaper and smaller version of the Arduino Uno, with the same processing power and functionality, and more analogue inputs
2. The ESP32: a low-cost, Wi-Fi enabled series of microcontrollers
3. The Raspberry Pi Zero W: the smallest Wi-Fi-enabled version of a raspberry pi.

Table 2 outlines the main specifications relevant to the sensing project, for each microcontroller.

Table 2: Microcontroller comparison table

	Arduino Nano	ESP32	Raspberry Pi Zero W
			
Cost	£4	£7	£10
Wi-Fi functionality	No inbuilt Wi-Fi, but can have external Wi-Fi module attached (£3). (Wifi-enabled Arduino's are much more expensive ~£30+)	Inbuilt Wi-Fi 802.11 b/g/n	Inbuilt Wi-Fi 802.11 b/g/n
Analog inputs	8 channel, 10-bit, can use 5V or 3.3V reference	18 channel ADC (but non-linear), 12-bit, 3.3V reference	No internal ADC
Digital IO	14 pins, 5V	39 pins, 3.3V	40 pin GPIO header, 3.3V
Programming method	Mini USB C++, Arduino IDE	Micro USB, Several programming options incl. Arduino IDE	External screen, keyboard and mouse, or use laptop with headless set-up, Python
SPI	Yes	VSPI, and HSPI	Two SPI interfaces
I2C	Yes	Yes	Yes, one standard and one EEPROM
Memory	Flash memory 32KB, SRAM 2KB	Flash Memory 4MB, SRAM 520KB	512MB RAM
Power Supply	6-20V unregulated, mini-B USB, or 5V regulated to pin.	Micro-USB, or regulated 3.3V or 5V	Micro-USB

Each of these are explored in a little more detail in the following sections. The Raspberry Pi Zero W was never tested, but is expected to perform similarly to the ESP32.

4.1 Arduino Nano

The Arduino Nano is a small, complete, and breadboard-friendly board based on the ATmega328. It has the same functionality as the popular Arduino UNO (the larger, more standard Arduino board), but in a smaller size. Arduino boards are widely used by students and hobbyists and ideal for projects with multiple inputs and outputs. There is lots of support available online making them extremely easy to work with and troubleshoot when learning. An Arduino Nano was used during initial testing, but several large limitations were found, leading to the decision to move over to the ESP32. The main challenges are described in this section. Some wiring diagrams for the Arduino are included in the report, as they may be transferable to other 5V logic microcontrollers in the future.

The Arduino has an inbuilt ADC, which although only 10-bit, is far superior in linearity when compared to the ESP32. Unfortunately, there are insufficient ADC pins to satisfy all the sensor requirements, so 2 Arduino's would be needed, or an external ADC chip. Due to the need for a SD card (SPI connection), and RTC module (I2C connection), connecting an external ADC chip which requires one of SPI or I2C could be challenging, unlike for the ESP32 which has two hardware SPI ports. It may be possible for both the ADC chip and SD card to be connected to the same SPI pins using a different chip select, but this was not attempted.

Another large downside is the lack of inbuilt Wi-Fi. Arduino modules with integrated Wi-Fi do exist, but they are about 5 times the cost (~£30). External Wi-Fi chips such as the ESP8266 (£3) can be installed to work with the Arduino for a total cost comparable to the ESP32. This was tested but proved unreliable and difficult to troubleshoot. More detail is explained in the Section 4.1.1.

The biggest challenge after integrated testing was due to the limited dynamic memory of the Arduino. When required to simultaneously write to an SD card, read sensors, and send data wirelessly, it ran out of dynamic memory and started performing unreliably. This was even without doing any data analysis to operate dump loads, and with fewer sensors than our system would require. It is possible the problem could be removed with more efficient programming, but in either case, the Arduino would be performance would be nearing its limits, which would not easily allow for future expansion and improvements.

4.1.1 ESP8266 Wi-Fi module

The ESP8266 communicates with the Arduino nano via software serial (the hardware serial is used by the computer during troubleshooting/programming). Software serial does not cope well at the default higher baud rate, so the ESP8266 should be reformatted to 9600 baud rate.

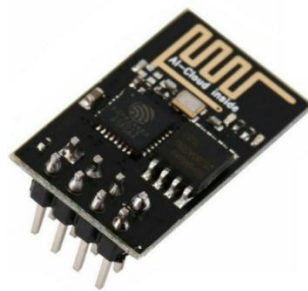


Figure 2: ESP8266 Wi-Fi Module

The Arduino cannot supply sufficient current through its 3.3V pin, unless powered from a 9V battery pack (it won't work when powered via USB). It was found to be able to supply sufficient power through its 5V pin via a step-down voltage regulator, but using an external power source is advisable. A wiring diagram is shown in Figure 3.

In general, although the ESP8266 was successfully tested with the Arduino, due to the difficulty in setting it up, and troubleshooting (error messages are difficult to 'catch' and send on to the computer serial port to identify the issues), it isn't recommended. Wi-Fi connections with the ESP32 were far easier to implement.

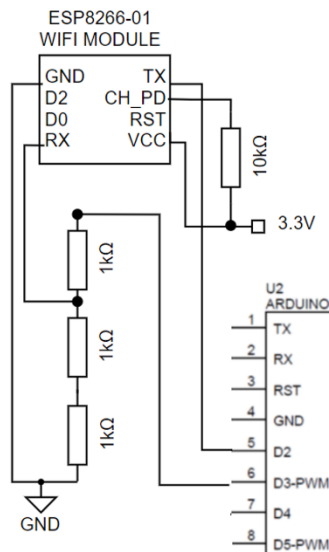


Figure 3: Wiring diagram: ESP8266 Wi-Fi module for Arduino Nano

4.2 ESP32

ESP32 is a series of low-cost, low-power 'system on a chip' microcontrollers with integrated Wi-Fi and dual-mode Bluetooth. The inbuilt Wi-Fi at a low-cost makes it more practical than the Arduino for use within a 'connected' monitoring device. The higher RAM (512kB) is also beneficial, as are the greater number of programmable pins (38), and the ability to go into low power mode in between measurements.

There are several development & break-out board variations which make them easy to work with. Each have slight variations on the chosen pins broken out (some have more, some have less), and different firmware flashed to the device. However, when programming, the original chip's pin is referred, not the breakout physical pin, so the same code should work with different ESP modules provided they are wired up correctly. The physical pin layout is identical for most boards with the same numbers of pins. It is often hard to tell when buying (especially from ebay) the exact ESP32 module being purchased, and it would appear that so long as the number of pins broken out on the module are the same, the boards can be used interchangeably. We used the following in our tests:

- 38-pin, AZDelivery, ESP32 DEVKIT V1 DOIT, ESP-WROOM-32, source: ebay
- 30-pin, JZK, ESP32 Development Board, ESP-WROOM-32, source: Amazon
- 30 pin, AZDelivery, ESP32-DevKitC NodeMCU, ESP32-WROOM-32D, source: Amazon

The ESP32 can be programmed in several different languages, frameworks, platforms and environments. For ease, we used the Arduino IDE, as learnings from the preliminary testing using the Arduino could be quickly and easily transferred.

Unlike the Arduino, many of the pins are programmable for different purposes. This means that although there is much more potential functionality for the ESP32, some of the stated functionalities cannot occur simultaneously. This has to be taken into account when designing the system to ensure clashes do not occur. For example, GPIO 0,2,4,12-15,25-27 should not be used if Wi-Fi is being used.

4.2.1 ESP32 ADC limitations and solutions

Although the ESP32 has an inbuilt ADC with higher resolution than that in the Arduino (12 bit vs 10 bit), the ADC has significant non-linearity as shown in Figure 4, and is unable to distinguish voltages at the lower and upper end of its range. This was verified with testing. Although the non-linearity could be corrected for in software, (assuming the sensors are optimised to only operate across the middle range of values as you would never be able to distinguish below 0.15V or above 3.1V), this would require more computing power than a simple linear conversion. Alternatively, careful hardware design could ensure that the signal range to be measured lies within the central portion of the ADC sensing range.

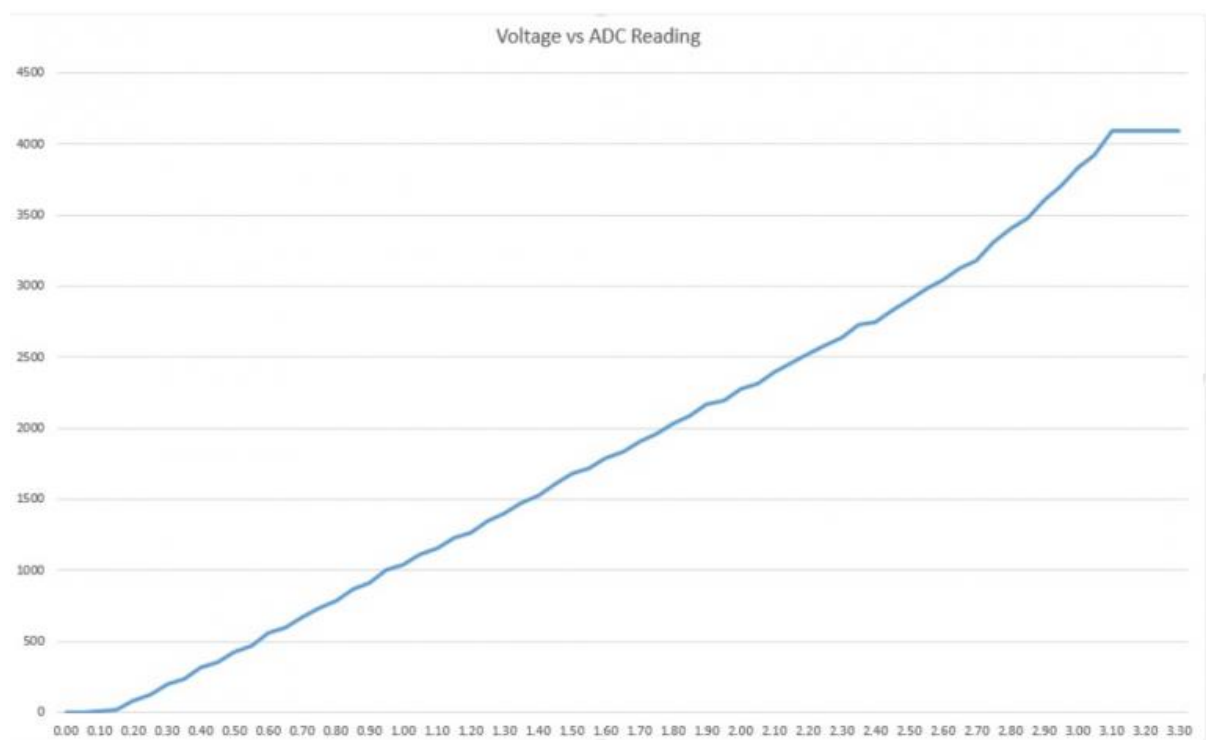


Figure 4: ESP32 ADC non-linearity. Voltage vs ADC reading

In addition, the second internal ADC chip is unable to operate whilst Wi-Fi is running, leaving only 6 channels on ADC1 for use which is not enough for all the required sensors. It would be possible to turn Wi-Fi on and off between readings, but this, combined with the non-linearity challenges, led to the decision to operate the ESP32 with an external ADC chip. These are available in a variety of formats, with different numbers of channels (e.g. 4, 8 or 10 channel), different resolutions (e.g. 10,12 or 16 bit), different communication protocols (SPI or I2C), and some with the ability to further amplify small signals (e.g. the ADS1015/ ADS1115 ADC chips). SPI ADC chips are preferable, as an I2C ADC chip would clash with the I2C used to communicate with a real-time-clock module, whereas the

ESP32 has two SPI interfaces, so one can be used for an ADC while the other is used to write to an SD card.

4.2.1.1 Testing results

The MCP3008 was chosen for testing. This chip is very similar to that found on the Arduino, with 8 channels at 10-bits. It communicates with microcontrollers via SPI, and is widely used in hobbyist projects so there are a lot of existing software libraries to make programming quicker and easier. Although it does not have enough channels for all our required sensors, the learnings and code should be easily adaptable for other 10 channel ADC chips that communicate via SPI. If necessary, two of the three AC voltage sensors can be omitted from the prototype, assuming a balanced AC load.

Standard Arduino libraries for the MCP3008 work on the ESP32 VSPI pins. To keep those available for SD card reading and writing, an alternative ESP32-specific library was found on GitHub which would allow use of the MCP3008 via the HSPI pins [1].

A wiring diagram is shown in Figure 5. Note that it was chosen to have VCC and VREF at 5V on the ADC module as many of the pre-built sensors operate at 5V logic. For that reason, a logic level shifter is needed before the ESP32 to bring the data lines down to 3.3V.

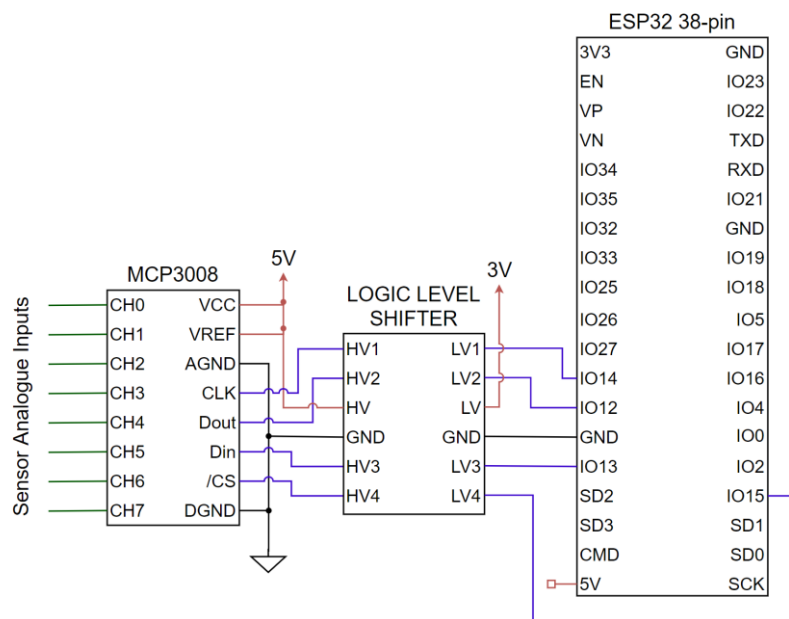


Figure 5: MCP3008, 10-bit, 8 channel ADC chip, wired to ESP32 via HSPI, through logic level shifter

GPIO12 is used as a bootstrapping pin to select the output voltage of an internal regulator which powers the flash chip (VDD_SDIO). This pin has an internal pulldown so if left unconnected it will read low at reset (selecting default 3.3V operation). [2] When connected as MISO to the ADC Dout, it cannot be pulled down, and flashing the ESP32 is not possible. The easiest solution is to unplug GPIO12 when first uploading code or powering up the device. However, for a final product, this is an additional undesirable set-up step. Alternatively, a command can be used to program flash voltage selection fuses to 3.3V. [2] To avoid permanently changing the underlying ESP32 code, it was decided to opt for a different HSPI wiring configuration, as shown in Figure 6.

The wiring was adapted from the SD card SPI connections found in [2]. ESP32 GPIO2 is also used as a bootstrapping pin and must be be low to enter UART download mode. This can be achieved by connecting GPIO0 and GPIO2 using a jumper, then the auto-reset circuit will pull GPIO2 low with GPIO0 when entering download mode. The preferred final HSPI wiring method is shown in Figure 6.

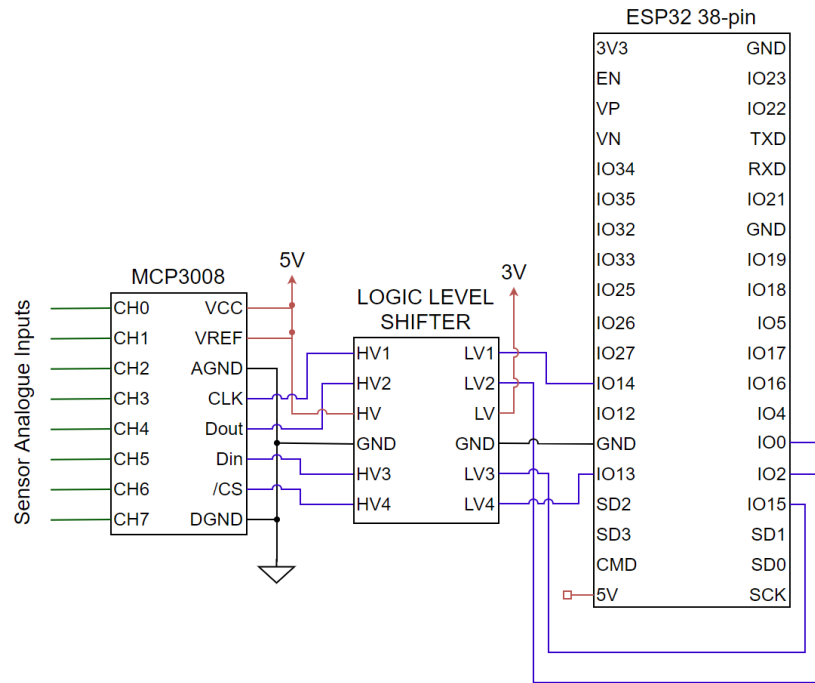


Figure 6: Improved wiring for MCP3008 and ESP32, via HSPI connection and logic level shifter

5 Sensing circuits

For each of the sensing circuits, the current or voltage to be measured is converted into a voltage between 0 and 5V (or 3.3V if using a 3.3V ADC reference) to be fed into an analogue to digital converter (ADC). Due to some of the sensing ICs being designed for a 5V output, it made sense to design all sensing circuits to operate at 5V such that the same ADC chip could be used for all.

In this section, the main design options explored are detailed, along with results of any preliminary testing. Note that further testing is still required at higher voltages and currents in the field, for evaluation on reliability and calibration.

5.1 DC Voltage

5.1.1 Potential Divider

5.1.1.1 Theory

The most straightforward method is using a potential divider as shown in Figure 7.

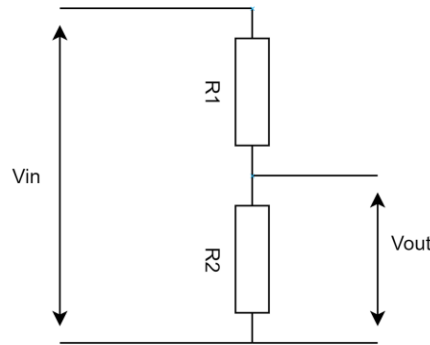


Figure 7: Basic Potential Divider for DC voltage measurement

The high input DC Voltage, V_{in} (e.g. from solar panels), is scaled down to a lower voltage according to the ratio of R_1 and R_2 . This lower voltage, V_{out} , would feed into the ADC of a microcontroller, or an external ADC, so it must not exceed the ADC reference voltage.

$$\frac{V_{out}}{V_{in}} = \frac{R_2}{R_1 + R_2}$$

In our application, we need to measure up to 1000VDC from a string of panels. Assuming a 5V ADC reference voltage, this requires:

$$\frac{5}{1000} = \frac{R_2}{R_1 + R_2}$$

To determine R_1 and R_2 , the following must be considered:

- 1) Acceptable total current draw and power losses
 - a. This is particularly important if the system is to be run off batteries.
 - b. $P = V_{in} \cdot I = V_{in} \cdot V_{in} / (R_1 + R_2)$
 - c. This determines the minimum total resistance of $R_1 + R_2$
 - d. Note that if $R_1 + R_2$ are too high, such that virtually no current flows through the divider, then accuracy of the reading can be affected. There must be a trade-off.
- 2) Standard resistor values
 - a. Resistors only come in certain standard values, so the exact ratio required may not be possible. R_2 should be rounded down to the closest standard resistor value to avoid overloading the ADC.
- 3) Resistor power rating and overload voltage
 - a. After sizing the resistors, the datasheets must be checked to ensure the power dissipated and voltage drop across each resistor does not exceed the rated values.
 - b. If necessary, R_1 , which has the largest voltage drop (and power dissipation) can be split into several resistors in series, reducing the voltage drop and power dissipation in each.
- 4) Physical distance if high voltages (1kV) are being used
 - a. High voltages can result in arcing if there is insufficient physical distance between the high voltage side and ground.
 - b. Splitting R_1 into multiple resistors can have the additional benefit of increasing the distance between the high voltage side and ground.

Resistors come with a tolerance (often 1% to 5%) which determines their accuracy to the stated resistance. Although this can improve immediate measurement accuracy without additional

software calibration, for best results, the sensor should be calibrated in the code after testing with high voltage measurements, so resistor tolerance is not too important.

High resistor values are necessary when working with high voltages to reduce power dissipated. However, the high resistance may affect the accuracy of readings:

- 1) High source impedance to the ADC causes a slowdown of the charging of the sample and hold capacitor inside the ADC. The sample can be taken before the settling time is reached, resulting in an inaccurate measurement.
- 2) ADC's have an input resistance, and a high source impedance forms the high side of a voltage divider, causing some attenuation of the input signal. Depending on the ADC, sources impedances greater than $1.1\text{k}\Omega$ could affect the accuracy of readings.

Two main solutions exist to the above:

- 1) For low bandwidth signals, adding a decoupling capacitor in parallel with the lower resistor can slow down the reaction time of the divider to a change in input signal, reducing the effective impedance seen by the ADC, and allowing it to make accurate measurements. Decoupling capacitors are also useful in removing high frequency noise, and removing current spikes. Example capacitor values range from $0.1\mu\text{F}$ to $1\mu\text{F}$. Often multiple capacitors are placed in parallel of different values to decouple different frequency ranges.
- 2) An operational amplifier has high input impedance and very low output impedance. Feeding the output of the voltage divider into an op amp in voltage follower (unity gain amplifier) configuration, will take the high source impedance as seen at the divider output, and change it to a low source impedance at the same voltage, for inputting into the ADC, hence reducing any attenuation issues within the ADC due to its internal resistance. Care must be taken to choose a 'rail-to-rail-input-output' (RRIO) op amp to ensure the output voltage is not scaled differently to the input.

In case of unexpected higher input voltages, to prevent damage to the ADC or any other components further down the line, a Zener diode should be placed in parallel with the lower resistor. Zener diodes have a rated reverse breakdown voltage. If this is chosen to match the ADC maximum rating (i.e. 5.1V for a 5V ADC), then in the case of a surge in input voltage, the diode will break down at 5.1V , providing a low resistance path for current to flow to ground, and preventing the voltage as seen by the ADC from rising above 5.1V . The Zener Diode breakdown is reversible, so once the input voltage returns to expected levels, the diode will not need replacing and will still protect the ADC from future voltage surges.

Figure 8 shows the updated generic recommended potential divider configuration for high DC voltage measurements, including a Zener diode, decoupling capacitor and voltage follower.

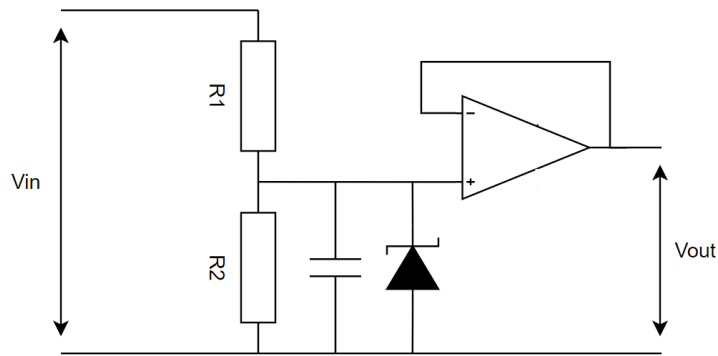


Figure 8: Potential divider for DC Voltage measurement, with voltage follower, decoupling capacitor and Zener diode

5.1.1.2 Final sensor configuration

In our solar installations, typical DC voltages from charge controllers are up to 50V, and up to 1000V from a string of solar panels. Using the same voltage divider for 50V as for 1000V would result in only the first few levels of the ADC being used, so precision is reduced (ie. a 2V error due to ADC precision is more acceptable when measuring 1000V than when measuring 50V). For this reason, it makes sense to separate the two DC voltage sensing circuits. The two chosen arrangements are shown in Figure 9 and Figure 10, calculated for a 5V ADC reference voltage. The resistor values could be changed for a 3.3V ADC reference voltage. A 5V reference was chosen so the same ADC could be used for other sensors operating at 5V. The calculations used to choose the resistor values are shown in the Appendix, Section 14.1.

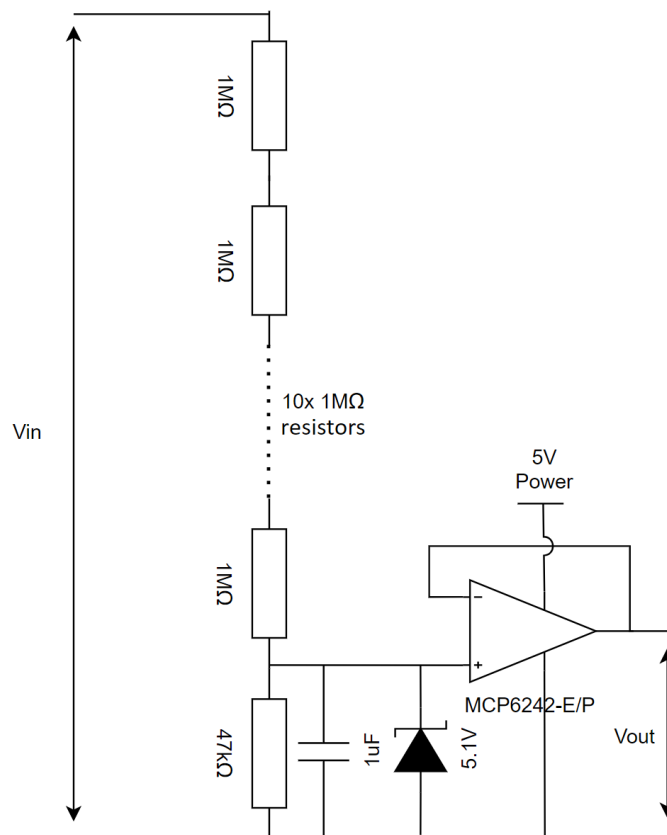


Figure 9: 1000V Voltage Divider – Final arrangement

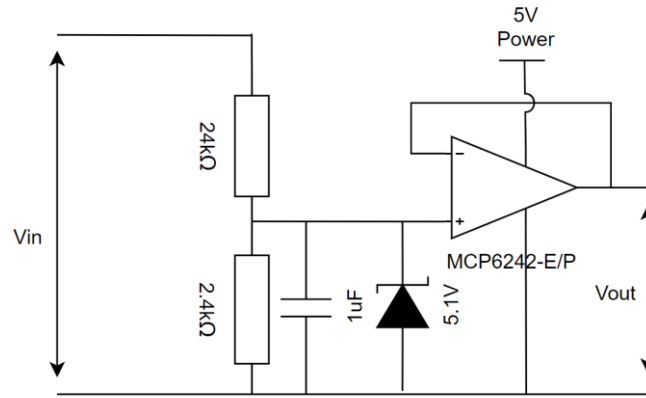


Figure 10: 50V Voltage Divider- Final arrangement

5.1.1.3 Test results

Detailed testing for a 250V maximum input potential divider was conducted. The following set up was used:

- $R1 = 50\text{M}\Omega$ and $R2 = 1\text{M}\Omega$ (50:1 ratio). The effect of lower resistor values was also tested, using $R1 = 10\text{M}\Omega$ and $R2 = 200\text{k}\Omega$.
- ESP32 using MCP3008 10-bit external ADC chip (with V_{ref} and V_{dd} at 5V)
- Voltage follower, using an MCP6242
- 5.1V Zener diode
- 0.1 μF Capacitor (a 1 μF capacitor was also tested)
- Logic level shifter (from ADC chip at 5V, to input into 3.3V logic level microcontroller)

Results were accurate to $\pm 0.5\text{V}$ even without the voltage buffer. This was surprising as research indicated the high resistor values should have caused errors within the ADC. It may be because the system was only tested up to 52V, when towards the upper end of scale, it may have resulted in larger errors. For maximum accuracy, it is probably best to include the voltage buffer when the source impedance to the ADC would otherwise be greater than $10\text{k}\Omega$.

Without a capacitor, the output would fluctuate around 4 ADC levels (1V total). Once a capacitor was added, the readings stabilised on one value, with only the occasional slip down one ADC level (0.25V). This was true both with and without a voltage follower.

The test results are shown in Figure 11.

Multimeter Voltage (V), Potential Divider Voltage (V) and ADC reading

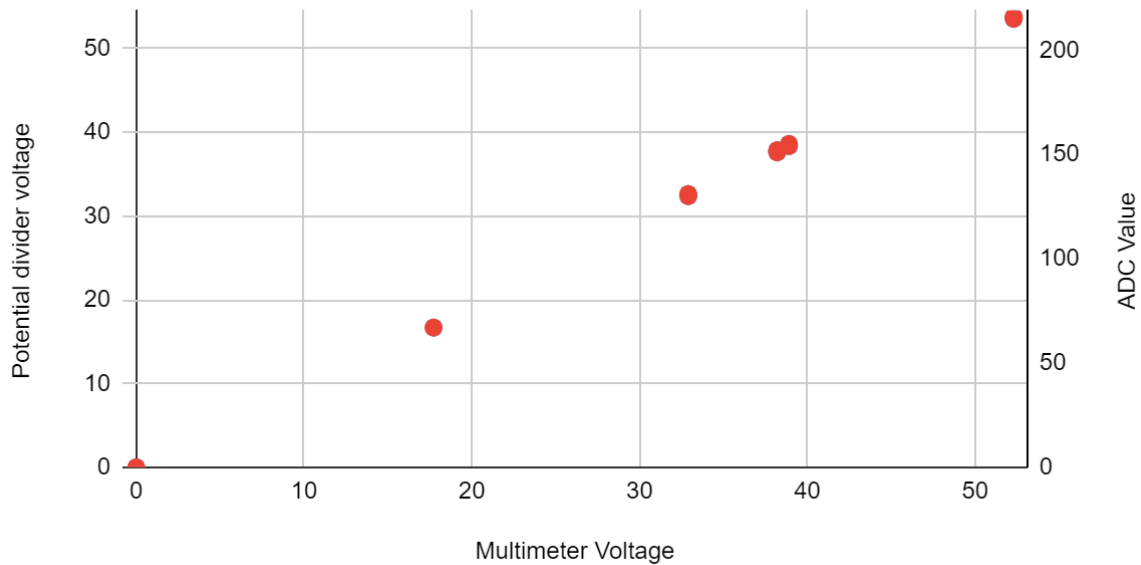


Figure 11: Graph of Multimeter voltages vs measured values during testing

5.1.1.4 Future improvements

The potential divider method is not an electrically isolated system. Although it is reliable, accurate, and cheap, from a safety standpoint the lack of isolation could be potentially dangerous, especially if a user interacts with the system, and there is some component failure, resulting in a high voltage being passed to the user. As user interaction is unlikely except to collect the SD card (a rare occurrence for which the power could probably be first disconnected), it is not essential, and has not been addressed in the first prototype. However, for future improvements, the high voltage AC and microcontroller should be electrically isolated. One method of achieving this would be using an optocoupler. This is explored in more detail in Section 5.3.2.3 for AC Voltages, although the circuit would be more straightforward for a DC linear mapping.

5.2 DC Current

5.2.1 DC Hall Effect Current Sensor

5.2.1.1 Theory

This is the safest and simplest way to measure current. The hall effect enables high currents to be isolated from the sensing circuit. Currents flow directly into the hall-effect sensor IC, generating a magnetic field that can be measured. Allegro microsystems produce a range of ICs which can sense either uni or bi-directional DC and AC currents, up to 50A, 100A, 150A or 200A. (Note that the current must flow directly into the chip, so for AC currents, the non-invasive split core current transformer is preferable for safety and ease of connection). A number of series of chips exist, but from examination of the datasheets, they appear to perform almost identically, producing a 0-5V output for measurement by an ADC. Any of ACS770, ACS772, ACS759, ACS773 or ACS758 can be used, depending on availability of suppliers.

When installed, it is best to measure current near the supply voltage of the load (the high-side) to bring immunity to ground bounces and allow for the detection of short circuits to ground.

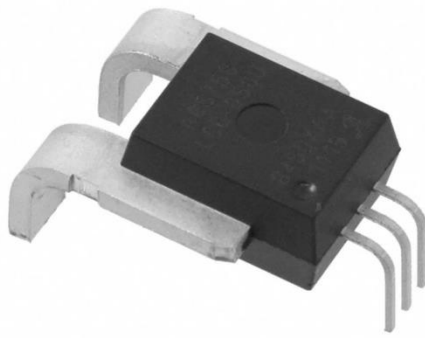


Figure 12: The ACS770 hall-effect current sensor

The Allegro microsystems hall-effect current sensors cost approximately £6 each. There is a zero-offset of 0.5V for the unidirectional sensor, observed during testing. For our systems, in which we only expect current to flow in one direction, the unidirectional sensors can be used. Unfortunately, the series of sensors do not go above 200A, and alternative higher-rated hall-effect sensors are about 10 times the cost. It was suggested the hall effect sensors could be installed in parallel, such that the current load in each is halved. This was tested and the results are shown in the following subsection.

The datasheet gives a recommended wiring diagram for noise reduction, as shown in Figure 13. R_F , in series with the output line, is believed to be for signal integrity (to avoid reflections due to mismatches in impedance of the transmission line) and current limiting (to ensure the maximum current that can be taken by the ADC is never reached). Suitable values range from 22.1Ω when used for signal integrity, up to 470Ω if current limiting is the main factor [3]. As the datasheet states the resistor is for noise management, a smaller resistor was chosen, of 33Ω . This is the same as the burden resistor used in the AC current sensor, hence reducing the variety of components required and reducing cost.

C_F is a decoupling capacitor, to remove high frequency noise. Values of decoupling capacitors are typically between $1\mu\text{F}$ to $100\mu\text{F}$ for low frequency noise, and $0.01\mu\text{F}$ to $0.1\mu\text{F}$ for high frequency noise. A value of $0.1\mu\text{F}$ was chosen, though this value could be tuned/improved.

C_{BYP} is a bypass capacitor to remove noise from the 5V supply. This is good standard engineering practice.

Application 1. The ACS770 outputs an analog signal, V_{OUT} , that varies linearly with the bidirectional AC or DC primary sampled current, I_P , within the range specified. R_F and C_F are for optimal noise management, with values that depend on the application.

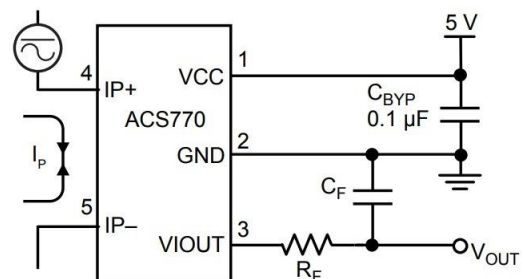


Figure 13: Recommended wiring diagram for ACS770, taken from ACS770 datasheet

Another source recommended $R_F = 4.7k$ and $C_F = 4.7nF$, and stating that they were not necessary when measuring DC Currents [4]. Another electrical design specialist should be consulted to optimise the resistor/capacitor values definitively.

5.2.1.2 Results of Testing

Due to the availability of resistors and capacitors at the time of testing, the final recommended resistor and capacitor values (R_F and C_F as shown in Figure 14) were not used. As these are for noise management only, the slight variations during testing are not believed to affect the measurements. Note that the ACS770 can be substituted for any of the Allegro series of hall-effect current sensors. Care must be taken so that the maximum current rating for the sensor chosen is not exceeded. V_{out} feeds into an ADC. A Zener diode is added for extra protection for the ADC. The high current circuit and sensing circuit are completely isolated (no common ground), making it extremely safe.

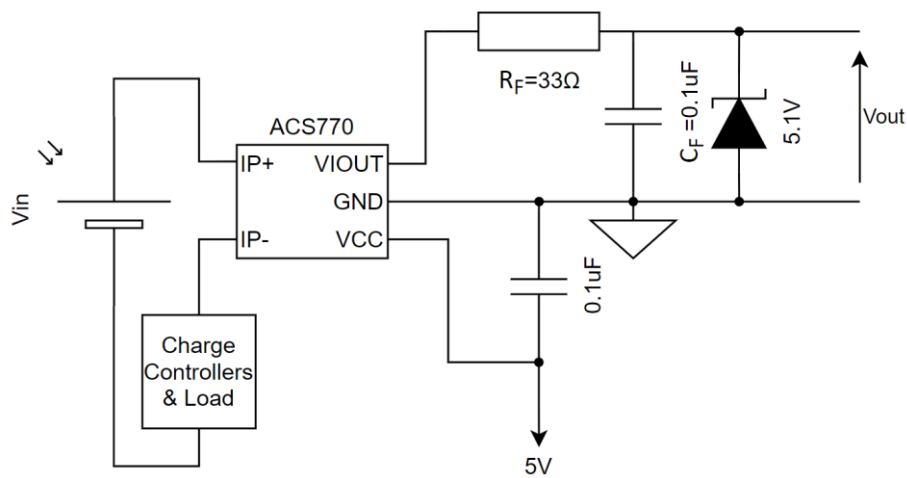


Figure 14: DC Current Sensing Circuit Wiring Diagram for Final Prototype

Figure 15 shows the results of testing, using $R_F = 270\Omega$, $C_F = 470nF$. Note that the ADC readings have not been converted into the equivalent current, however, when this was done with calibrated values, the error was minimal ($<0.2A$). Testing was done using an ACS770-50U sensor, and repeated with two different ACS758-50U sensors. It is interesting to note that although the response of all sensors were linear, the response was slightly different, especially between the ACS770 and ACS758. This highlights the fact that software calibration is needed for each device before/during final installation, and if two sensors are installed in parallel, the same sensor type should be used.

50A DC Current Sensor performance

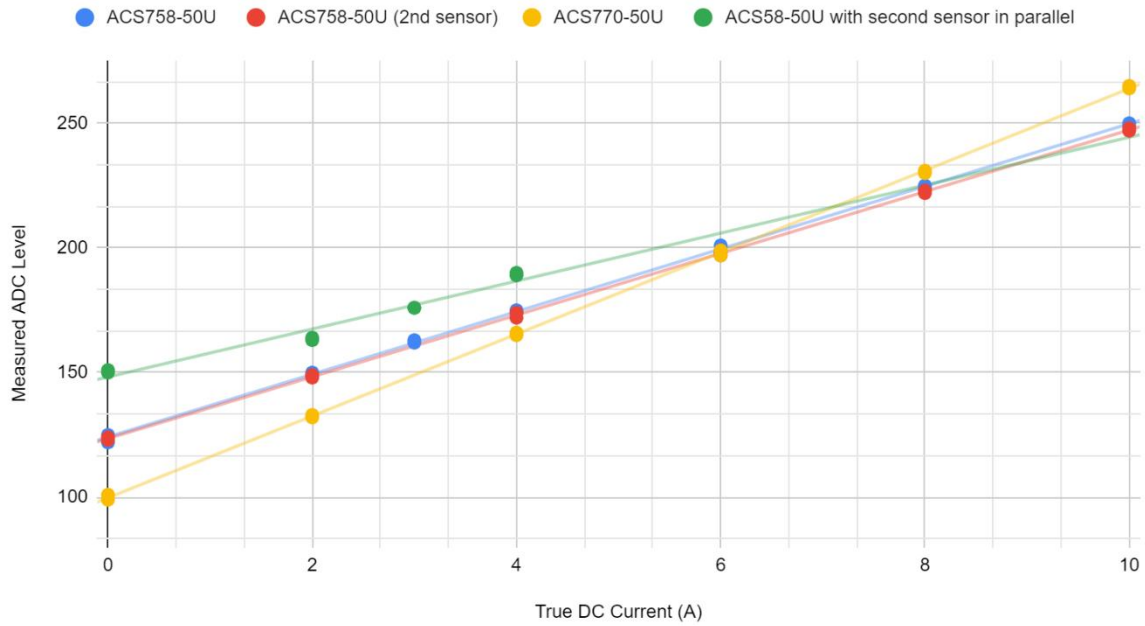


Figure 15: ACS758-50U and ACS770-50U DC Current Sensor Testing results, $R_F=270\Omega$, $C_F=470nF$

To test the feasibility of installing two 200A rated hall effect current sensors in parallel to measure up to 300A, two ACS758-50U sensors were arranged in parallel as shown in Figure 16. (The lower rated sensors were used due to the max 10A test currents available, and it was assumed the learnings could be applied to the higher rated sensors). The measurements taken by one of the parallel sensors is also shown in Figure 15 in green, plotted against the expected current running through the sensor (ie. a 10A input current should result in 5A running through each sensor). As can be seen in the graph, this resulted in a slightly different and less linear response. The different response may also be due to poor connection as the two legs were crocodile-clipped together for testing. Within the testing range, the error remained below 0.4A, but testing at higher currents is required as only the very low end of the range could be explored with the available equipment.

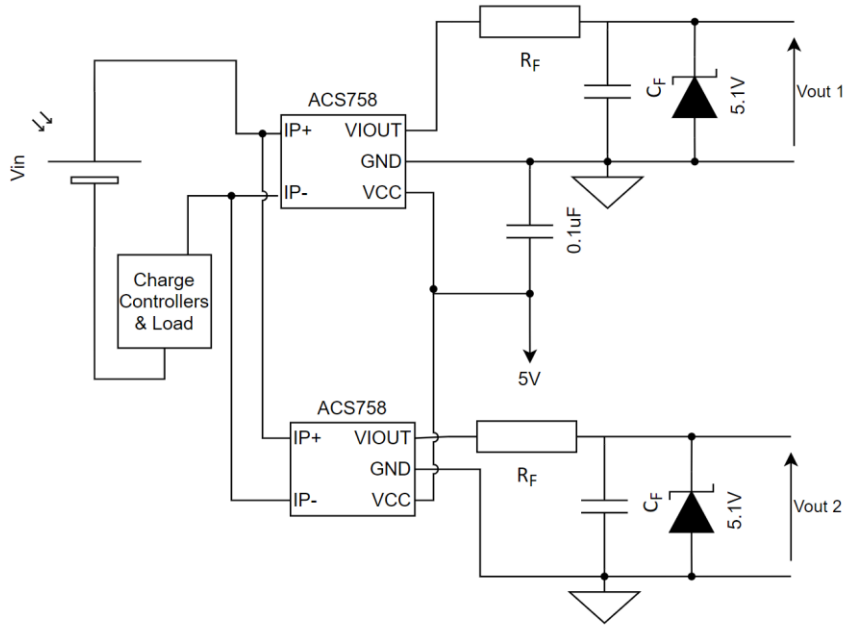


Figure 16: Wiring diagram showing two DC current sensors installed in parallel, to halve the current through each. During testing, the values $R_F=270\Omega$, $C_F=470nF$ were used.

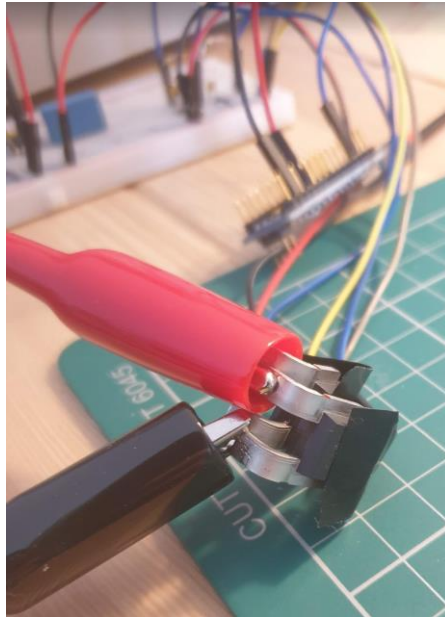


Figure 17: ACS758-50B wired in parallel directly to power source during testing

The ACS770-100U and ACS770-100B were also tested while in Tanzania, using $C_F = 10pF$, $R_F = 4.7k\Omega$. Due to the thickness of wire required for high currents, significant twisting forces can be transferred to the current sensor when attaching the wires to the large metal contacts. This resulted in the three thin data signal legs of the sensor to be broken off for three sensors. In practice, care must be taken to ensure strain relief is placed on the wires and no force is transferred through to the breadboarded/soldered legs which are weak and liable to breaking.

The testing currents fluctuated significantly when measured with the multimeter in Tanzania. The extremities at each datapoint are plotted in Figure 18 which shows promise for a linear result, although a comment on accuracy cannot be made due to the variation in the 'true' comparison value.

It is expected that all sensors in this range have a highly linear, accurate response, as shown in the tests from the 50A sensor.

Multimeter Reading (A) and Sensor Reading (A)

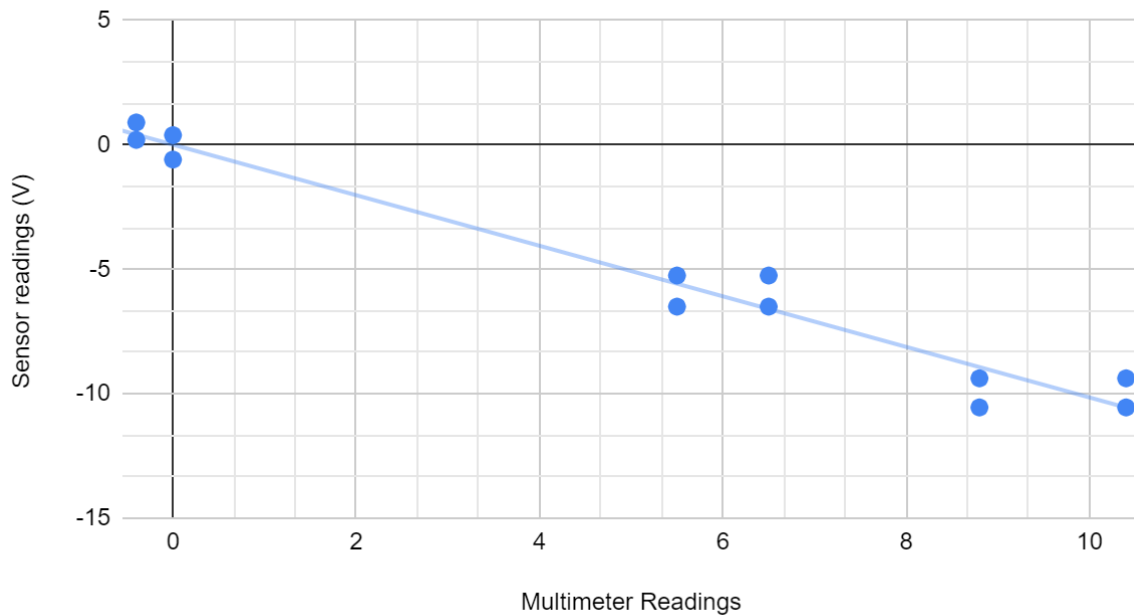


Figure 18: Multimeter Reading vs Sensor Reading for 100A bi-directional hall-effect current sensor. $C_F = 10\text{pF}$, $R_F = 4.7\text{k}\Omega$

5.2.2 Current Shunt

A current shunt is an alternative method of directly measuring current using Ohms Law $V=IR$. The shunt resistor is placed in series with the circuit load and the voltage drop measured. In order not to affect the load current or dissipate excess power, the resistance of the shunt is extremely low. This results in a very low voltage drop, and so additional amplifiers are required.

Although this method was researched, it was decided that the hall effect sensors would be a better choice, due to their low cost, high accuracy, low complexity (no additional amplifiers required), and most importantly from a safety aspect, the electrical isolation they provide between the high voltage/current to be measured, and the low voltage & current sensing circuit.



Figure 19: Current shunt example. Wires are screwed into each side of the metal block.

5.3 AC Voltage

For a 3-phase balanced system, the voltages for each phase should be equal. From a measurement of one phase, the other phases can be inferred. In star configuration, the line-to-line voltage (VLL) is related to the phase voltage (VLN) by the equation $VLN = VLL / \sqrt{3}$. However, in practice, due to unbalanced loading, or faulty motors, the phases may be unequal, meaning that separate sensors for each phase are preferable.

Knowledge on power consumption of the load may be useful. In order to calculate the apparent power (VA), we need only multiply the RMS line voltage (V) by the RMS phase current (A). However, this doesn't really give any useful information on power usage. To calculate the true power of a circuit, we need to sum the instantaneous voltage multiplied by instantaneous current over one full cycle. Depending on the load type (resistive, or reactive), the true power may be negative at points during the cycle. Therefore, knowing the sign of the voltage, or the phase difference between the current and voltage, is extremely important to be able to measure the real power being consumed.

There are several methods for measuring AC voltage, some of which use diodes which result in the loss of the negative half of the waveform. The different methods and their advantages/disadvantages are outlined in the following section. For all methods providing a full sinusoid, it was assumed the Open Energy Monitor library: EmonLib, would be used for measuring the voltage and calculating power directly from the sinusoidal signal. The library needed to be rewritten to allow for use with an external ADC, as it was made for directly reading the analogue inputs on an Arduino [5].

5.3.1 Voltage Transformer

5.3.1.1 Theory

This is a highly recommended approach, used by most of the projects in the Open Energy Monitor resource. A voltage transformer is used to step down the high AC voltages, and a potential divider fine-tunes the output to the 0 to 5V range as required by the ADC. Two resistors are used to bias the output such that the sinusoidal waveform is centred around 2.5V. This is explained well by Figure 20, taken from the Open Energy Monitor.

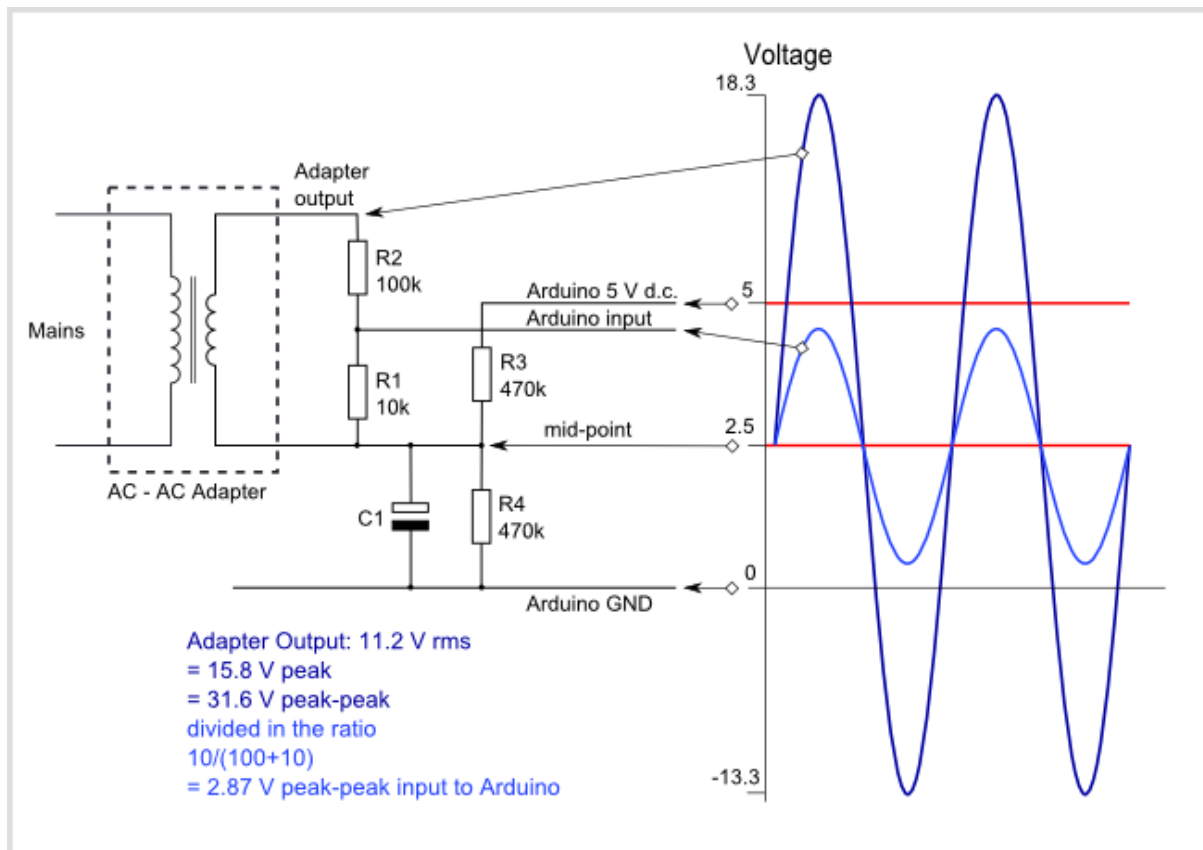


Figure 20: AC Adapter method for measuring high AC voltages. C1 is anything between 1uF and 10uF

A large advantage of the voltage transformer method is that the high voltages are electrically isolated from the microcontroller. When using the Open Energy Monitor biased output design, the full voltage sinusoid is passed to the ADC, so true power calculations can be made.

The ZMPT101B is the most popular voltage transformer module among hobbyists, which contains all of the above features, with some extra diodes and protection, all for ~£6. The potential divider is implemented through the use of a potentiometer, allowing the output voltage to be scaled/calibrated to either 5V or 3.3V as required. The transformer is rated to 500VAC, but the module is only rated to 250VAC, most likely due to some of the additional safety components included. There is some phase shift introduced by the module, but this is consistent and the EmonLib library allows for calibration to remove errors in power calculations.

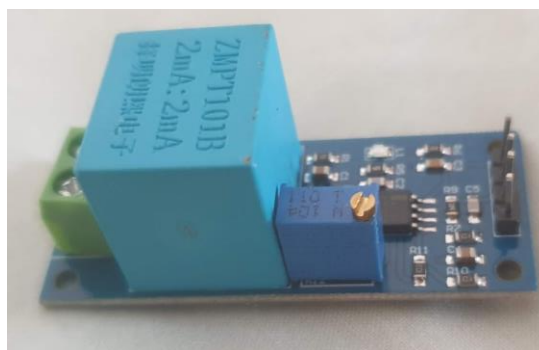


Figure 21: ZMPT101B Voltage Transformer Module

Unfortunately, the ZMPT101B module cannot measure voltages above 250V, making it impractical for our specifications (up to 500VAC), but the module was tested to help further our understanding

and verify the code, with the intention of then building a similar voltage transformer circuit with appropriately rated transformers and resistors. Further research later illustrated that finding through-hole PCB transformers with a primary winding voltage rating between 264V and 1000V is extremely difficult, so this method may no longer be appropriate. The transformer used in the ZMPT101B module can only be found from unverified Chinese websites and is hard to source independently of the module. Due to the inconsistency of the datasheets, it is not recommended that the transformer be repurposed from the more widely available ZMPT101B module. This module and method can therefore be used only in systems which go up to 250V AC.

Transformers rated up to higher voltages can be found as DIN Rail Panel Mount Transformers. These are typically rated 208 to 600V AC on the primary winding, converting this to 0-12 AC on the secondary. This could then be fed into a biased potential divider, to convert the signal into a smaller, positive signal for the ADC. This method would be extremely safe, but significantly more costly (At least 6 times more, as the DIN Rail transformers are £43 upwards).

5.3.1.2 Testing Results

The ZMPT101B module was wired directly into a programmable AC inverter output, allowing the voltage to be varied between 210V and 245V. As the EmonLib library could not work directly with the ESP32, the module was first tested with an Arduino for sensor accuracy. Several adapted GitHub libraries were found for interfacing EmonLib with an external ADC, but initial testing found some of these to be unreliable, and it was important to be able to reliably evaluate sensor accuracy during initial testing, so an Arduino was used.

In the first test, the EmonLib calibration value was left at the default software value of 234.26, and the potentiometer adjusted until the sensed (and calculated) voltage matched the multimeter reading. The inverter voltage was then adjusted over the full range and the error observed. At 245V, the error was -5V, and at 210V, the error was +5V. This significant drift over a relatively small voltage range indicates the calibration method is probably not appropriate. The response seemed linear, as can be seen in Figure 22, but there appeared to be a zero-offset that the calibration method has not accounted for.

ZMPT101B vs Multimeter voltage

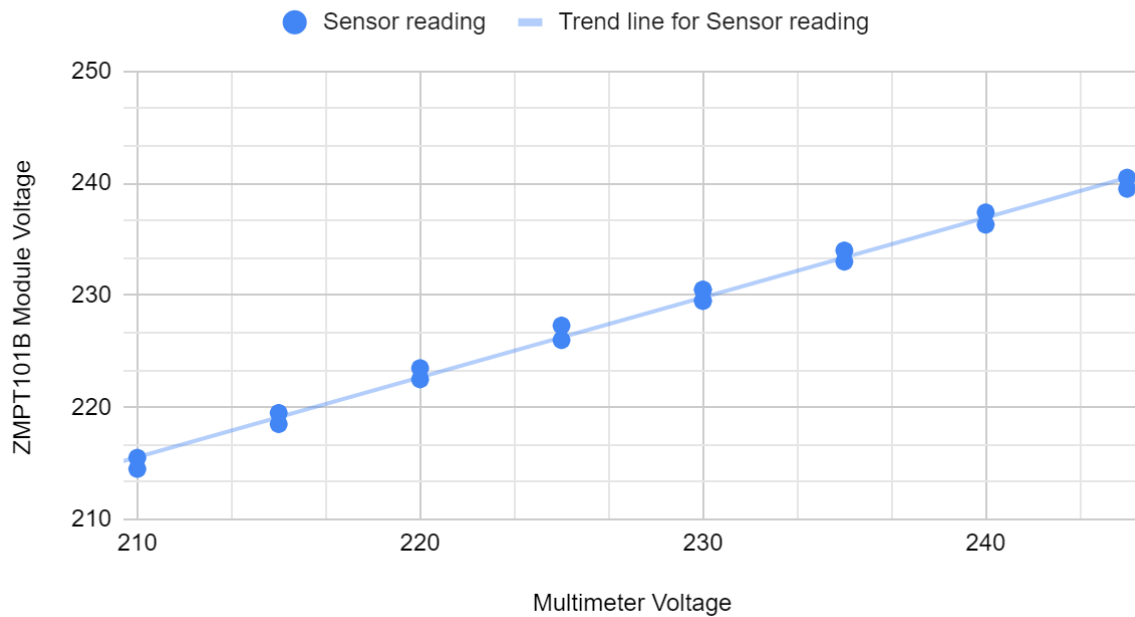


Figure 22: ZMPT101B measured voltage vs Multimeter voltage when calibrated at 230V

To improve on accuracy, a different calibration method was attempted. The sinusoid output from the ZMPT101B was viewed at 245V, and the potentiometer tuned to until the sinusoid covered the full ADC range without distorting (without exceeding 5V). The EmonLib software calibration value was then adjusted until the calculated output voltage matched the multimeter reading of 245V (this gave a calibration value of 282V). The results of the testing are shown in Figure 23. The accuracy is much improved, with an error of only -1.2V at 210V, the data point furthest from the calibration value. It is thought that in the first test, as the raw sinusoid was not examined, there may have been some clipping of the waveform, which resulted in unreliable calculations of the RMS voltage.

ZMPT101B measured voltage vs Multimeter voltage

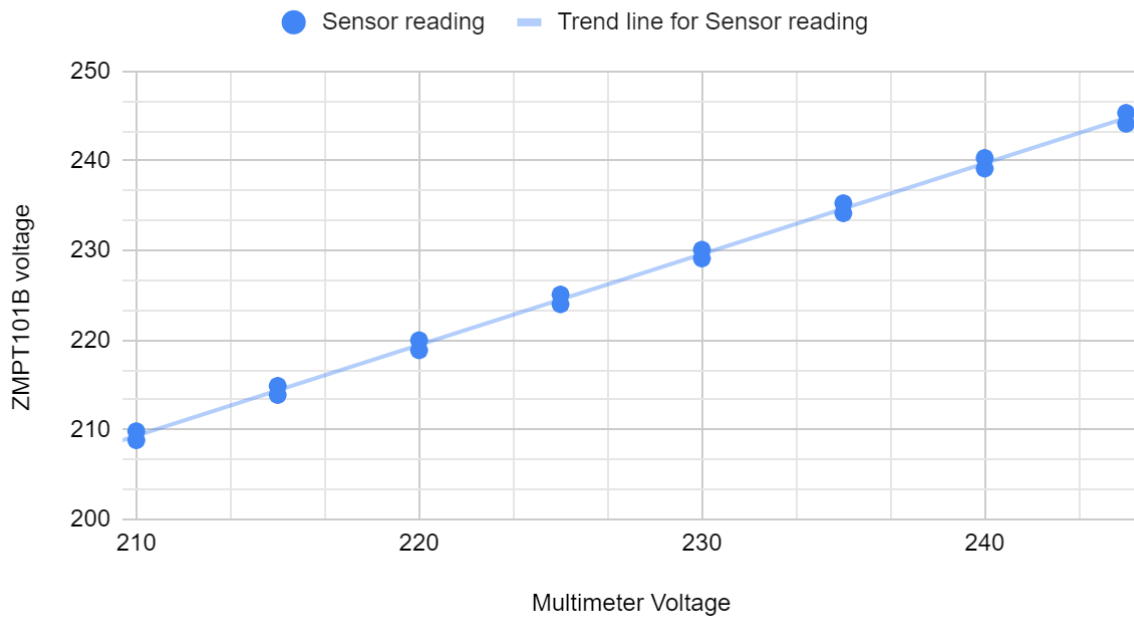


Figure 23: ZMPT101B measured voltage vs Multimeter voltage when calibrated at 245V, ensuring no waveform clipping

If used in the future, the sensor should be tested at lower voltages to see if the accuracy extends down to 100V AC. Unfortunately, these voltages were not available at the time of testing.

5.3.2 Potential Divider

5.3.2.1 Method 1

In the same way as for a DC voltage sensor (see Section 5.1.112), high AC voltages can be scaled down to a lower voltage for input to an ADC using a potential divider, as shown in Figure 24. The ratio of R_1 to R_2 will determine the size of V_{out} . Unlike in the DC version, a diode rated for high AC voltages (e.g. IN4007) is added following V_{in} to ensure the output voltage is always positive and will not damage the ADC. This means that only half of the voltage waveform is captured by the ADC, so the EmonLib library cannot be used directly, and calculating true power will be more complex, requiring some inferring of voltage values. For this reason, alternative solutions were sought preferentially.

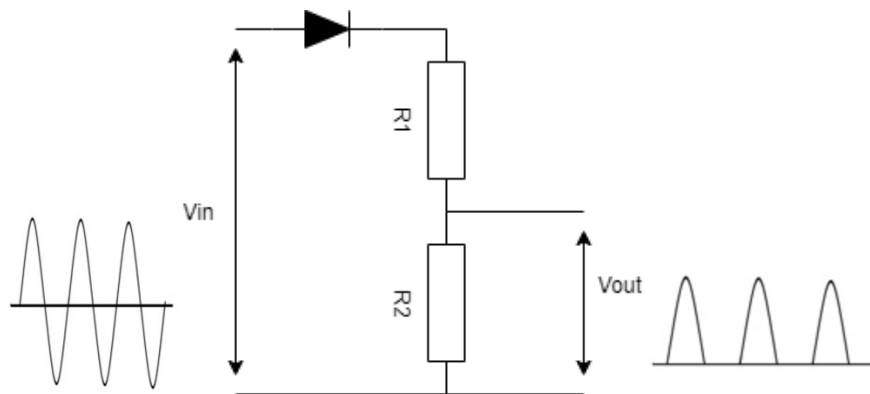


Figure 24: Potential divider for AC voltage measurement

5.3.2.2 Method 2

To avoid needing to cut half the AC sinusoid to avoid feeding negative voltages into the ADC, the output can be biased, as in Figure 25. The circuit is taken from [6].

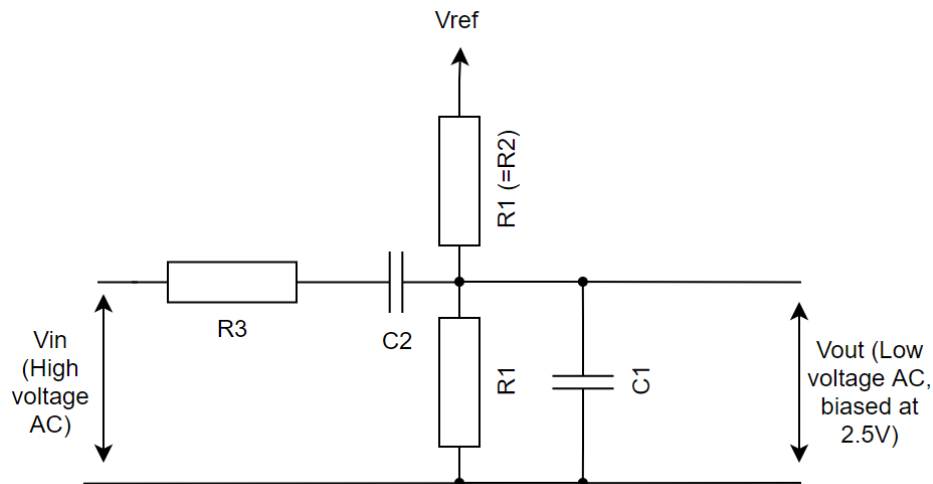


Figure 25: Potential Divider method for full sinusoid

The following bullet points explain the key design features. The circuit uses small signal theory:

1. R1 and R2 act as a parallel resistor load when referenced to R3, creating a voltage divider with R3 and $(R1 \cdot R2) / (R1 + R2)$.
 - R3 must be rated for the AC input voltage and expected power dissipation. It is recommended to use at least two resistors in series, each rated for the full AC input voltage.
2. R1 and R2 lift the ADC input voltage by $V_{Ref}/2$
3. C2 enables the ADC voltage to be shifted by $V_{Ref}/2$. It is a high pass filter. The voltage across C2 is usually 1.5V DC, about no AC. For safety reasons it is recommended to use an X2 foil capacitor rated to the high AC input voltage
4. C1 is a low pass filter to avoid high frequency noise at the ADC inputs

Following calculations for the values of capacitors and resistors (see Appendix, Section 14.1.3), the circuit shown in Figure 26 was determined, to take an AC voltage input up to 500V RMS, and output an AC analogue signal centred on 2.5V, with maximum amplitude 2.5V.

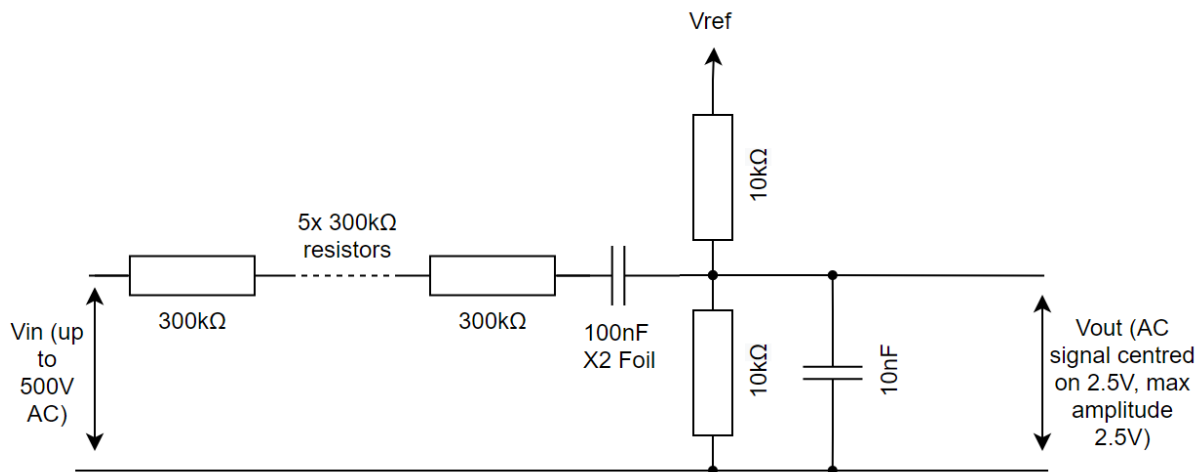


Figure 26: Potential divider circuit to get full AC sinusoid. Input up to 500V RMS, output centred on 2.5V, with a 2.5V max amplitude.

5.3.2.2.1 Testing results

The circuit from Figure 26 was breadboarded and tested using a variable AC voltage transformer as the input to obtain a range of AC voltages for testing and calibration. An Arduino Nano was used for preliminary testing, to verify the accuracy of the sensors, as it was known that the EmonLib library could not work accurately with an external ADC as required for the ESP32 (this would be fixed in software later). Due to the availability of resistors in stock at the time of testing, instead of 5 x 300kΩ resistors, 4 x 200kΩ and 2 x 360kΩ resistors were used, although the principle remains the same. The calculations are included in Appendix, Section 14.1.3.

The variable transformer produced voltages from 0 to 300V AC. (The X2 foil capacitor used was also only rated to 305V AC, so testing did not exceed this value, although assuming appropriately rated components, the system should work for higher voltages up to 500V AC.

Due to the high resistor values, a voltage follower was added before input to the Arduino ADC, to ensure there was no interference with the internal ADC resistors. (A more detailed explanation is available in Section 5.1.1.1, as the same theory applies to both DC and AC voltage dividers). However, testing revealed no difference with and without the voltage follower.

The AC waveform was first analysed using the Arduino serial plotter (reading from the analog input every cycle, and printing to the serial port), to confirm the waveform was indeed centred around 2.5V and that the amplitude appeared to increase and decrease as the input voltage varied. If the resistors are sized incorrectly, clipping of the waveform can occur, as shown in the final waveforms in Figure 27. This was seen to severely affect performance, resulting in a non-linear mapping, and errors of +40V as opposed to +1V.

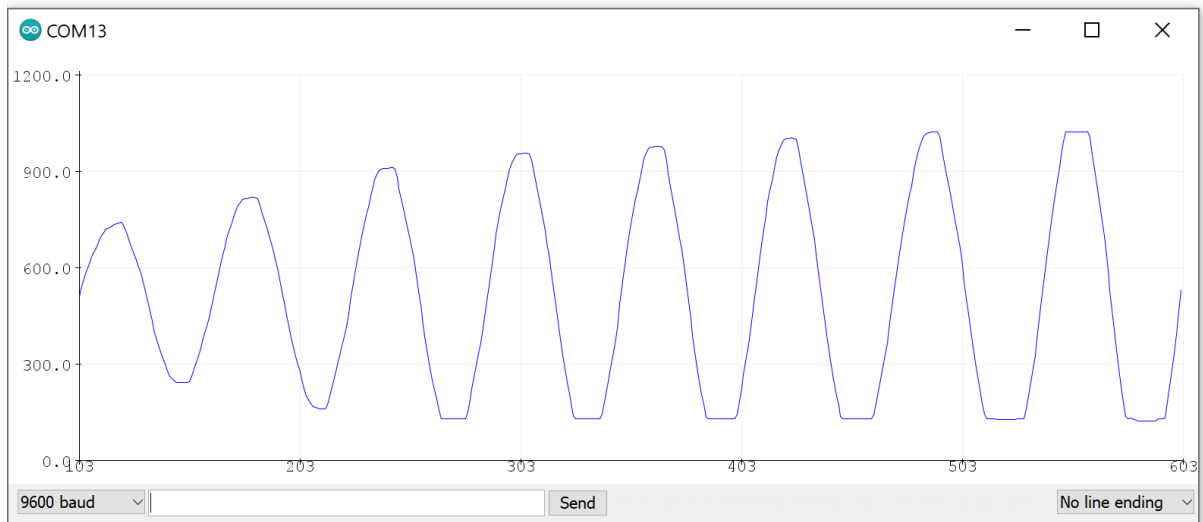


Figure 27: Example ADC output, shown on Arduino serial plotter. Here, incorrect resistor values had been used, resulting in clipping of the signal as the voltage was increased

The waveform was then analysed using the EmonLib software to calculate the RMS voltage, adjusting the calibration value until the measured RMS value and multimeter reading matched at around 300V AC. The practical calibration value, found to be 300, was very close to the theoretical calibration value of 305 (the expected input AC voltage per 1V output). The discrepancy can be attributed to resistor tolerances. The voltage was then varied and the two measurements compared, as shown in Figure 28. The divider was found to have excellent accuracy, better than $\pm 1V$ over the full range of values tested.

Voltage divider sensed voltage (RMS) vs Multimeter measured AC Voltage (RMS)

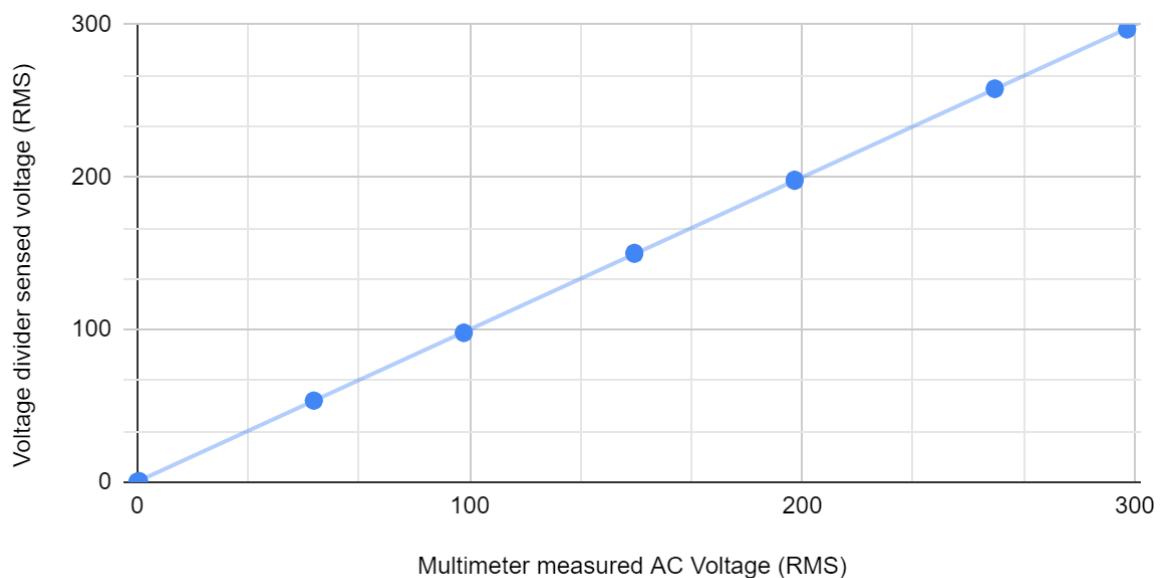


Figure 28: Biased AC Potential divider reading vs Multimeter reading

5.3.2.3 Future Improvements

In the same way as the DC voltage divider method, this sensing system is not isolated from the microcontroller. This is highly advised, especially if there is any type of user interface connected to the microcontroller (even a laptop), as electronic components can sometimes fail, and in the worst case, high voltages could be passed to the user. Due to time constraints, this was not implemented here, but should be included in the future.

An isolating analogue optocoupler is one method of achieving this. It would go after an initial potential divider, and before any voltage biasing that is linked to the ADC's voltage levels. (See Figure 29) When using an optocoupler, the ground levels on the optocoupler's input and output side are not connected. This allows the ground planes of both sources to be different. It is generally not recommended to connect the ground planes from different sources as this can result in electrical glitches and be hazardous.

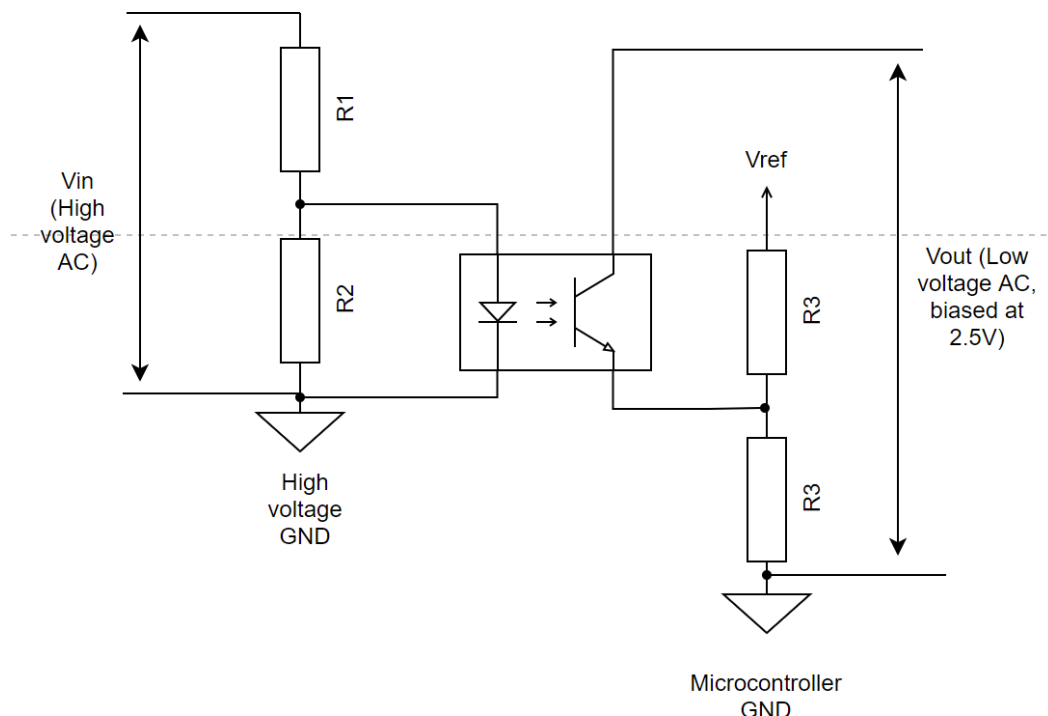


Figure 29: Simplified diagram showing concept for AC voltage potentiometer and isolating optocoupler, followed by DC voltage biasing ($R3$)

Good linear examples of DC optocouplers are the HCNR200 and IL300, costing around £5-£8 each. Note that they can only take a positive input, and so 2 must be used in a more complex circuit with op amps in order to map an AC signal. Although AC optocouplers do exist, a suitable linear mapping one was not found (they are typically used as digital logic systems to detect a high voltage, and so do not give an analogue mapping). More research may help source a suitable optocoupler.

Figure 30 shows an example bipolar isolation amplifier from the HCNR200/201 datasheet [7]. More detail on sizing resistors is available in the datasheet. It uses two optocouplers, OC1 and OC2. OC1 handles the positive portions of the input signal and OC2 handles the negative portions. Diodes D1 and D2 help reduce crossover distortion by keeping both amplifiers active during both positive and negative portions of the input signal. Balance control $R1$ adjusts the relative gain for the positive and

negative portions of the input signal, gain control R7 adjusts the overall gain of the isolation amplifier, and capacitors C1-C3 provide compensation to stabilize the amplifiers. The gain resistors would need resizing for our high AC voltages, and the ground side of the output would be fed into a biasing potential divider, such that the waveform centres around $V_{ref}/2$.

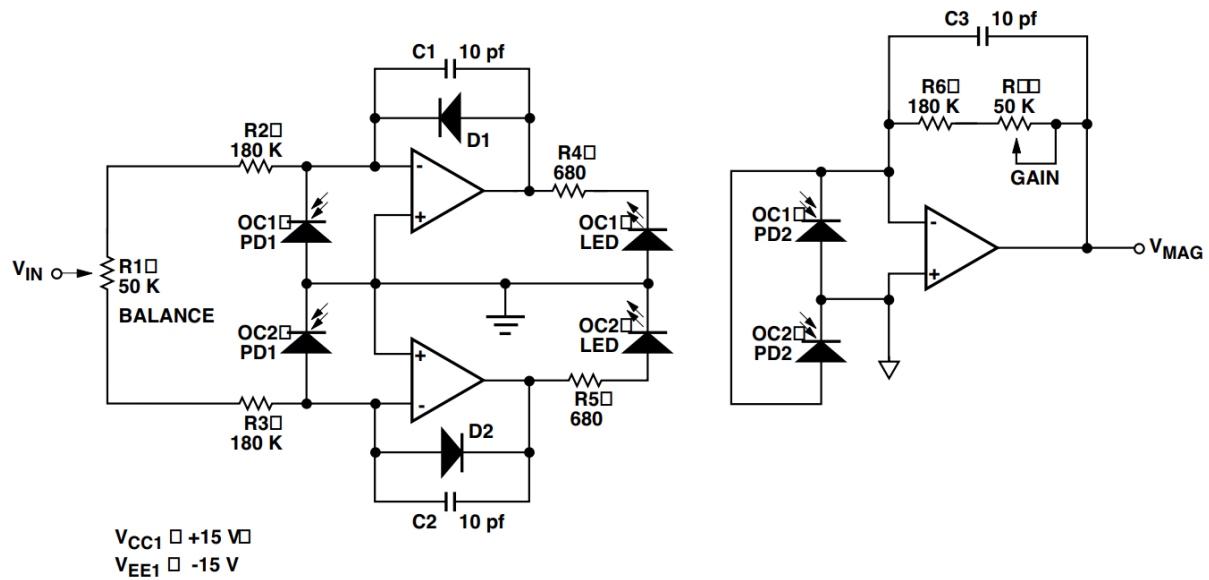


Figure 30: Example bipolar isolation amplifier, taken from HCNR200/201 datasheet [7]

To avoid the complexities and errors introduced from mapping an analogue signal, it is also possible to isolate the ADC from the microcontroller. However, this would require isolating all data lines, *and* the 5V power, V_{ref} , and ground, introducing additional complexities. (These would need to be converted from the high AC voltage supply).

Another example was found from [8] to interface analog signals from one circuit to another by setting up a standing current through the LED and then modulating this current with the analog signal (see Figure 31). In this example, the input to the op-amp is biased at half-supply volts (R1 & R2), and this is removed after the op-amp (C2). This method still has the challenge of getting a reference voltage on the isolated side of the op-amp, and in our use-case, we would not want this removed for feeding into the ADC.

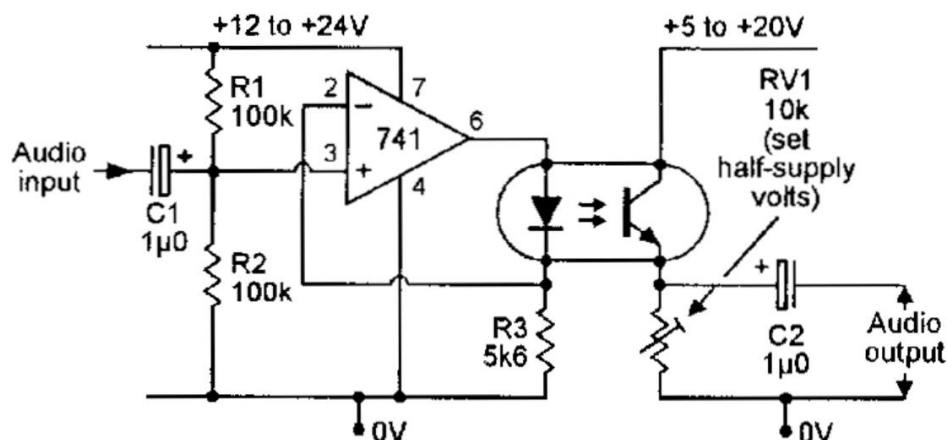


Figure 31: Audio-coupling circuit [8]

5.4 AC current

5.4.1 Split Core Current Transformer

5.4.1.1 Theory

The most straightforward method to measure AC current is to use a split core current transformer. This is a non-invasive, isolated method, and does not require splitting any existing AC wires. A split core current transformer can be bought for around £7. Different ratings are available (e.g. 30A or 100A), and the output voltage can be adjusted by choice of burden resistor. The Open Energy Monitor project [9] has lots of detail on sizing burden resistors, and installing the sensor. For a 3-phase system, three transformers can be used, one for each phase.

Figure 32 shows the wiring method for connecting a CT sensor to an Arduino analogue input (or external ADC for the ESP32). The sensor is clipped over the current carrying wire to be measured.

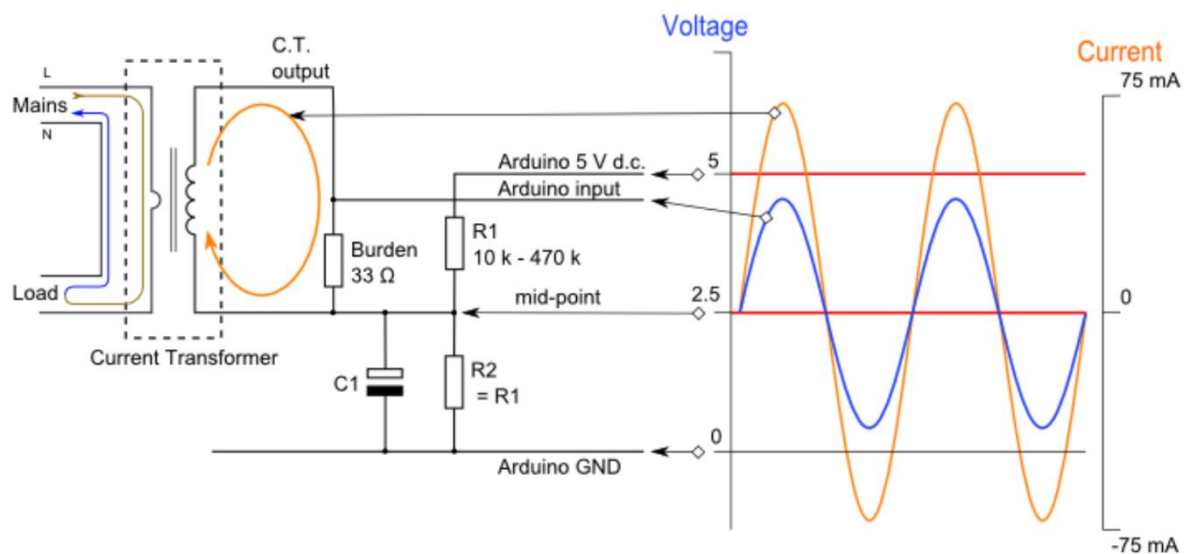


Figure 32: Split Core Current Transformer (CT sensor) wiring diagram for Arduino ADC. Adapted from [9]

R1 and R2 bias the output of the CT sensor around $V_{ref}/2$. Any value between 10k Ω (for mains powered applications) and 470k Ω (for minimum power consumption in battery powered systems) would be suitable. C1 provides a path for the alternating current to bypass the resistor. A value of 10 μ F is suitable.

Note that some CT sensors, e.g. the ARCELI 30A SCT-013-030, have inbuilt burden resistors, fixing the output voltage range. For these sensors, the burden resistor is simply omitted, leaving no direct connection between the two CT outputs. For sensors terminated in a 3.5mm headphone jack, the plug tip acts as the 'earthy' side, and the plug sleeve as the input signal to the ADC.

For CT sensors without a burden resistor, care must be taken never to place the CT over a wire without first either short circuiting or placing a resistor between the outputs. Unsafe voltages would be produced at its leads which could break down the insulation and destroy the C.T. Most transformers have internal Zener Diodes to limit the maximum unburdened voltage output to safe levels.



Figure 33: Split Core Current Transformer

5.4.1.2 Testing

The EmonLib [5] library was used to measure the RMS current from the raw analogue input. Due to the difficulty in interfacing EmonLib with an external ADC, as required for the ESP32, the sensors were tested using an Arduino Nano, and the inbuilt analogue pins, to guarantee software reliability.

The ARCELI 30A SCT-013-030 sensor was first tested, giving the results as shown in Figure 34. A calibration value of 30 was used with EmonLib, representing the calibrated 1V output for 30A through the sensor. As can be seen, only currents up to 4A were available, and the results measurements were only accurate to around $\pm 1A$. More testing would be needed near the higher end of the current range.

Multimeter Reading (A) vs CT Sensor Reading (A)

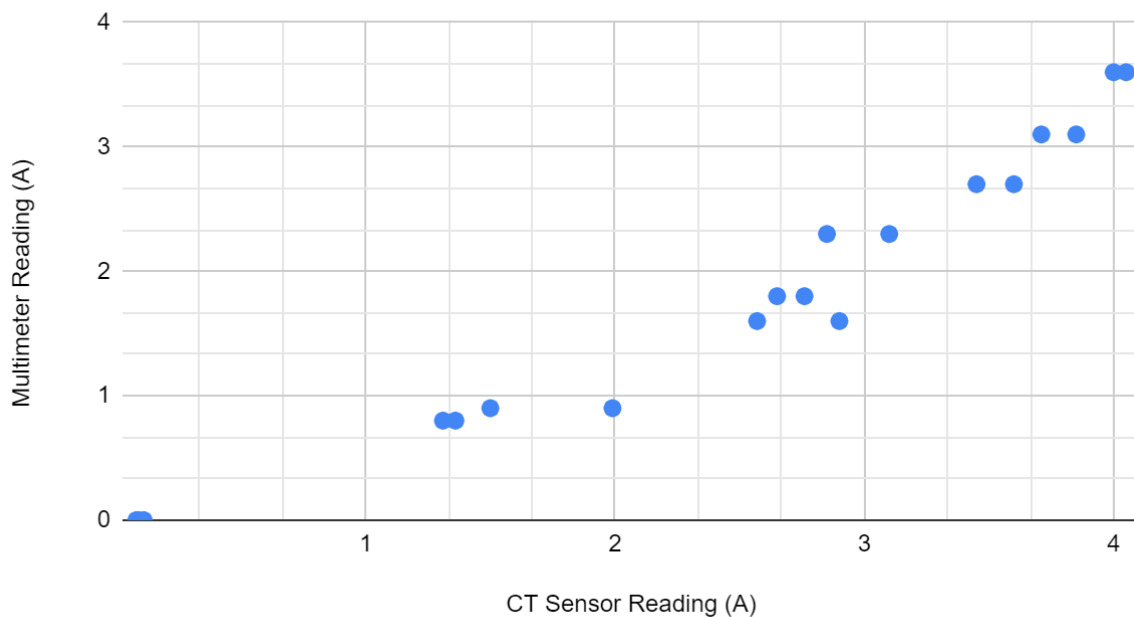


Figure 34: ARCELI 30A SCT-013-030 sensor test results

For the higher currents up to 100A, the ARECELI SCT-013-000 100A CT was used, with a burden resistor of $33\ \Omega$ to give an output of 4.67V at 100A. This gives a calibration value of 60.6. More detail on the calculations can be found in the Appendix, Section 14.1.4. Again, only currents up to 4A were available for testing, so the accuracy of the sensor over the full range can not be commented on. The initial tests show a consistent underestimate of current. This may be due to the resistor not being

exactly $33\ \Omega$, meaning the calibration value would need changing. Although the results do not show very high accuracy, due to the wide usage of these sensors in home-energy monitoring systems, it is believed that with appropriate calibration at higher voltages, this method should be reliable and accurate for the final monitoring system.

Multimeter Reading (A) vs CT Sensor Reading (A)

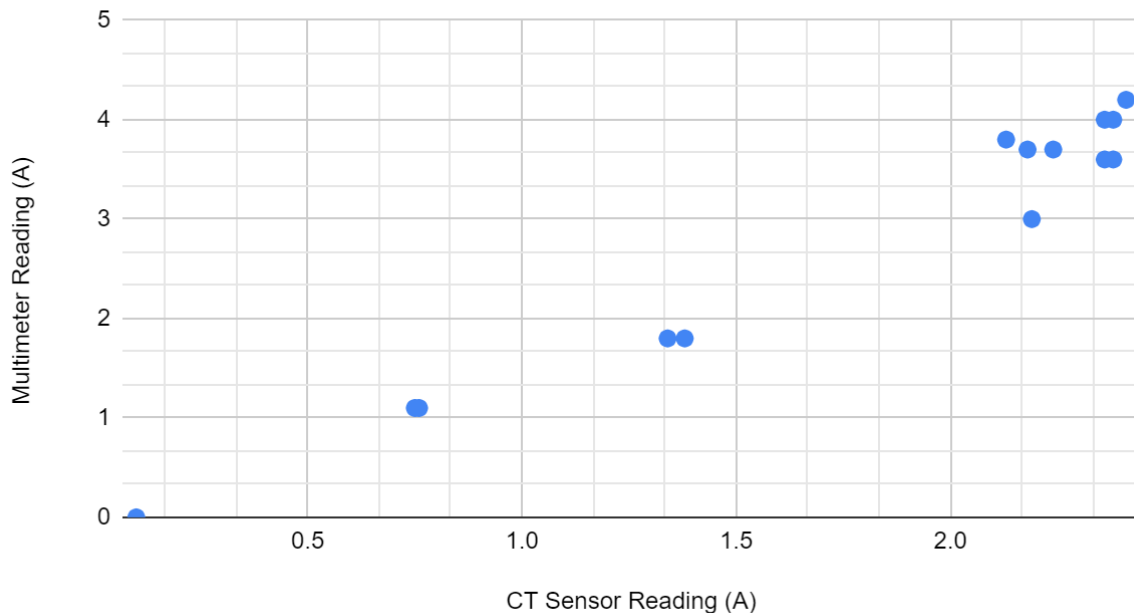


Figure 35: ARCELI SCT-013-000 1000A CT Test results

5.5 AC software library

As EmonLib directly reads values from an analogue pin, this makes it incompatible with an external ADC. Several users have created forks of the library to enable use with an external ADC, but some of these were found to be highly inaccurate. The code adapted by Paul Wieland [10] which incorporates a customisable call-back function was found to be most reliable.

When tested with AC voltages up to 300V, the AC voltage waveform was first examined in the Arduino IDE Serial plotter, showing the scaled amplitude was behaving as anticipated and with no waveform clipping. However, the calculated voltage using the forked library resulted in a much lower estimated voltage. Increasing the calibration value from 301 (as calculated based on resistor values, and proven to work with the Arduino) to 465, enabled the same accuracy to be achieved. It is not clear why this error occurs, and whether the call-back function slows the sampling rate, hence skewing the readings. More investigation is needed, but so long as the calibration constant is carefully tuned, this library is still useable.

6 Data Logging

6.1 Local Storage – SD Card

6.1.1 Theory

Both the Arduino and ESP32 can communicate with an external SD card reader to write data locally. The SD card module connects to the microcontroller using the 4-pin SPI interface (CS, MOSI, SCK,

MISO), as shown in Example 3.3V SD-Card module wired directly to a microcontroller with an SPI interface, and 3.3V supply.

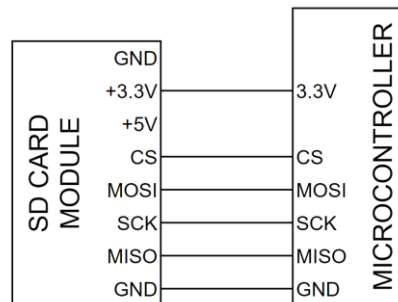


Figure 36: Example 3.3V SD-Card module wired directly to a microcontroller with an SPI interface, and 3.3V supply

Note that due to the prevalence of hobbyist Arduino projects, which operate at 5V logic, most SD card and micro-SD card reader modules also operate with a 5V supply and 5V external SPI logic, using internal voltage regulators (the SD card itself operates at 3.3V). SD card modules operating at 3.3V are also available, though typically are harder to find, with longer lead times. Many 5V SD card modules also claim to work with 3.3V logic (though the supply must be maintained at 5V), however it unclear whether they step-down the output voltage for the microcontrollers. Other 5V boards give no mention of being 3.3V logic-level compatible. Although 5V logic does work when forced into the 3.3V ESP32 GPIO pins, this is not recommended and could cause long term damage.

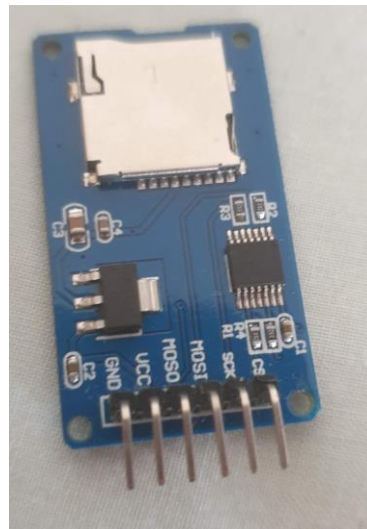


Figure 37: 5V Micro-SD Card Module

6.1.2 Test Results

A 5V micro-SD card module with an Arduino via SPI, and a 3.3V SD card module with an ESP32 via VSPI connection, were both successfully tested. See Appendix, Section 14.3 for the software code to write to an SD card, using the Arduino libraries, <SPI.h>, SD.h and FS.h. The specific wiring for the ESP32 is shown in Figure 38, and for the Arduino in Figure 39. The 5V micro-SD card module was also shown to work with the ESP32, although the specifications indicate it was probably outputting a 5V logic level which is likely to damage the 3.3V logic level GPIO pins on the ESP32 if used long-term.

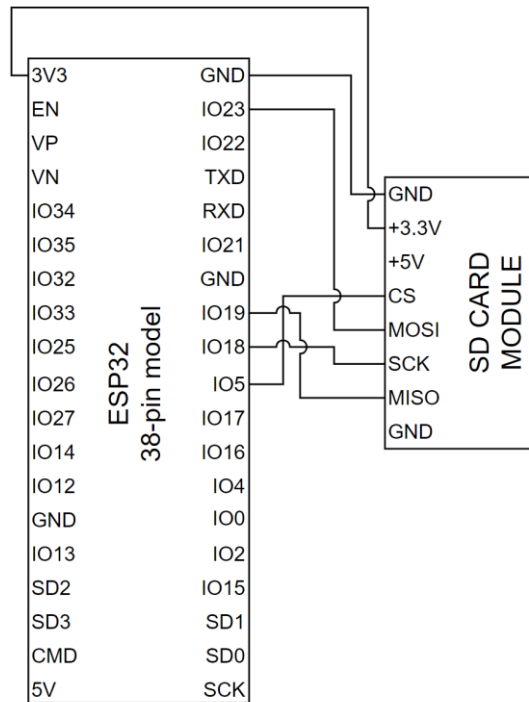


Figure 38: SD Card module wiring to ESP32 via VSPI connection

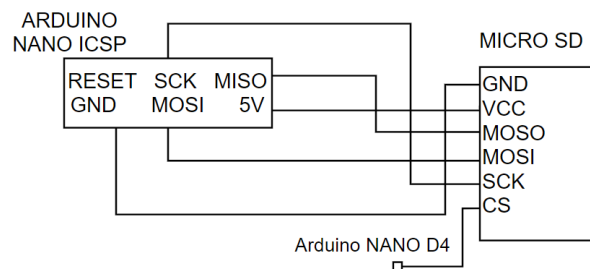


Figure 39: Arduino Nano Micro SD Card wiring diagram using the ICSP header for all connections except CS

Both SD card modules were found to be extremely sensitive to power losses. At first it was thought the ESP32 could not consistently supply sufficient power for the read/write process while running other functions. An external power source was used which improved reliability, but showed that if power was lost even momentarily during the program, any future communication with the SD card would fail. More investigation led to the decision not to use an additional power source, as the reliability was equally variable for both methods. More research needs to be done to understand why writing to the SD card will not work even after power is restored unless the entire program is restarted. As restarting the program can fix the issue, this indicates the failures could be fixed in code, by 'restarting' the SD card every time it is written to.

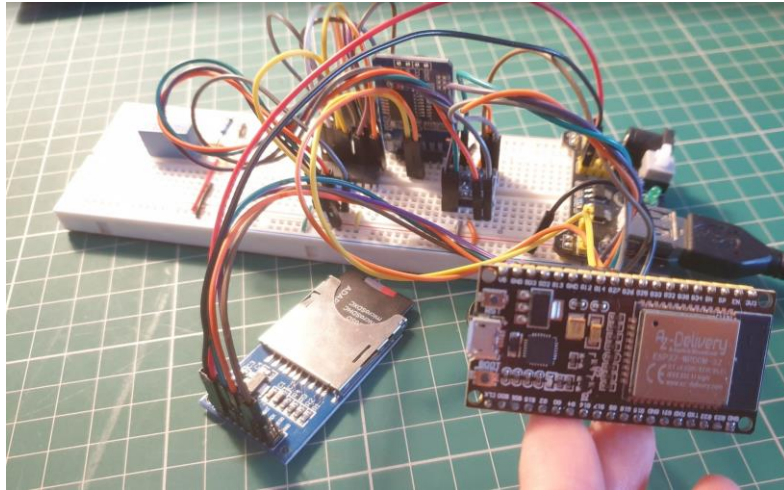


Figure 40: Breadboarded SD card and RTC module with ESP32 and external power supply

Note that these SD and micro SD card modules cannot handle high-capacity memory cards. Usually, the maximum identifiable capacity of the modules is 2GB for SD cards and 16GB for microSD cards. Assuming 10 data fields, each logging of data could take up to 100 bytes. For the smallest SD card capacity (2GB), this would allow 20 million data points. Sampling every 10 seconds, data could be stored for 6 years, before the SD card runs out of memory.

6.2 Writing data to the cloud

There are several options of servers that can receive data. The option that was researched in most depth was using ThingSpeak, as this seemed the most straightforward for Arduino projects. However, in the future, it is believed the other monitoring options should be investigated, as their user interfaces may be better. Examples include:

- Home-assistant.io
 - Open source home automation software. Can link multiple different connected devices, and display them all on a server. It can also be used for automations within the home assistant (most commonly used for home monitoring, (e.g. detecting doorbells, measuring temperature, operating lights etc.)

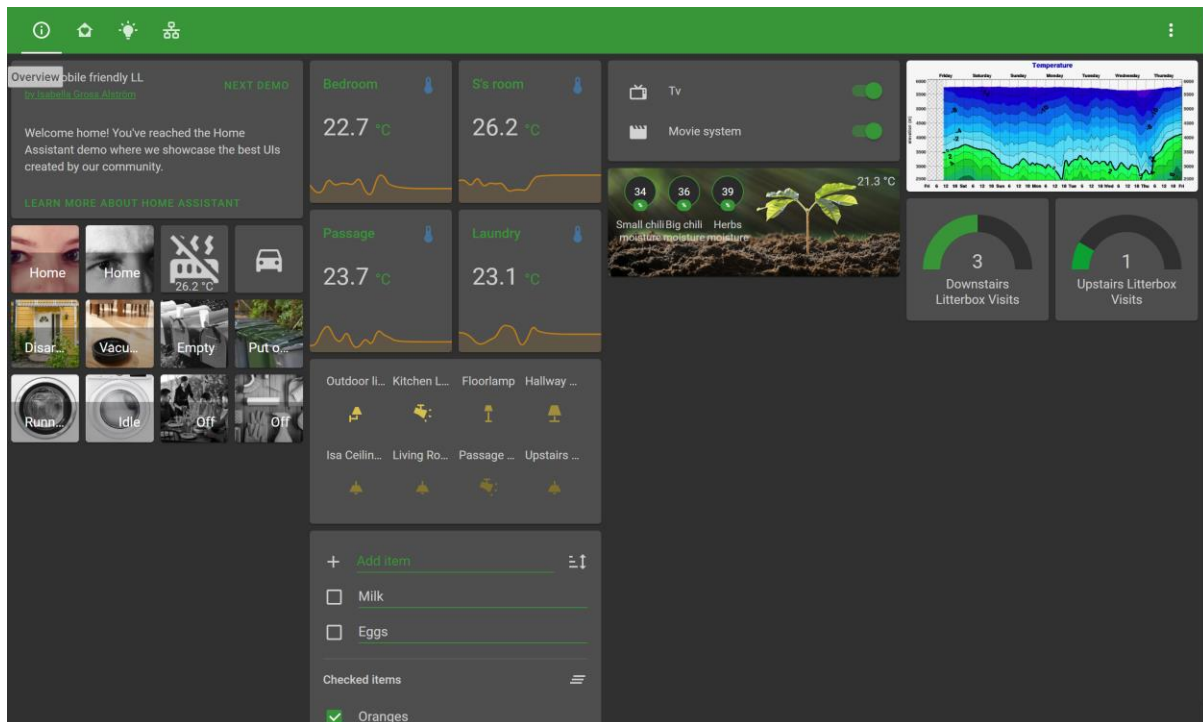


Figure 41: Home-assistant.io

- Emoncms.org
 - Developed as part of the OpenEnergyMonitor.org. Costs £1 per data channel per year (or can host the cloud service on your own raspberry pi, but it is easier to pay for them to manage it)
 - Open-source web-app for processing, logging and visualising energy, temperature and other environmental data.
 - Limited to one data point every 10 seconds.



Figure 42: Demo of emoncms.org

- Amazon Web Services (AWS) IoT service, DynamoDB, GraphQL API

- This was one online hobbyist approach to data collection. See [11] for more information.
- Uses AWS to send data to AWS IoT service which writes the raw reading to a DynamoDB table.
- Every night, a Lambda function archives all readings from the past day to a single CSV file
- Data visualisation using a GraphQL API that runs on Lambda, allowing user to query the latest readings or readings between a given time range.

6.2.1 ThingSpeak

This offers free data storage and analysis of time-stamped numeric or alphanumeric data. Data is sent to the ThingSpeak server where it is displayed on graphs online, and the data can be downloaded in CSV format, although the titles of the data fields cannot be saved, so the user must remember the order in which data is being sent over. It is only free for up to 3 channels per user, each channel containing 8 data fields, a latitude, longitude and elevation, and a 'status' (a text message for additional data). If more data is needed (e.g. 10 sensors, and a dump load status), any one of the fields or the status field could be sent as text, so that data from two sensors is sent together with a comma delimiter ("SENSOR1 , SENSOR2"). This could be easily separated out in excel. Alternatively, data could be written to multiple channels within the same program (e.g. one channel displaying AC data, and one channel displaying DC data).

There is also a limit of 3 million messages per year. This equates to 5.7 messages per minute, or 1 message every 10.5 seconds. ThingSpeak fields can only be updated once every 15 seconds using the free option. A message is defined as a write of up to 8 fields of data (including status, and location data). Each message cannot exceed 3000 bytes. Free users are limited to 10 million messages of data storage, which is 3+ years at the minimum update interval. There are commercial options available to increase the data storage, and sampling frequency.

6.2.1.1 Testing Results

ThingSpeak was successfully used to write data to the cloud, with the ThingSpeak library. As received data is automatically given a timestamp, it is unnecessary to send a time-stamp with the data.



Figure 43: ThingSpeak Online Data Visualisation example. Note that the status is not displayed online, but can be seen in the downloaded csv file.

6.3 Keeping Time

When taking sensor measurements for later analysis, having an accurate time-stamp is essential. For data transmitted wirelessly, often the method of receiving will have a time stamp on the data. A time-stamp is still required for data being written to the local SD card. If Wi-Fi is available, then the date and time can be requested using the Network Time Protocol (NTP). In case Wi-Fi is unavailable, the system should also have a method of remembering the time locally. Both the ESP32 and the Arduino do not have an inbuilt real-time-clock (RTC) module. They count time from the moment a program first begins, which is unhelpful especially if there is an unexpected power outage, and the system restarts at an unknown time. For this reason, an external RTC module is preferable, powered with a small battery to keep track of time under any circumstances.

Each of the two methods are explained in the following sections.

6.3.1 NTPClient over Wifi

This is achieved purely through software, so long as there is a Wi-Fi connection available. An online tutorial was followed successfully, showing that this could be a viable option if the Wi-Fi connection is known to be stable [12].

6.3.2 RTC module

There are several RTC modules available, the most popular being the battery powered DS3231, and DS1307. Both are functionally the same, except that the DS3231 has a built-in temperature compensated crystal making it more accurate, and uses less power, lasting a minimum of 8 years without an external 5V power supply. The DS3231 is however about double the size of the DS1307. They both use the same I2C address and software libraries can generally work interchangeably on either module. When not running off battery power, the DS3231 can operate at 3.3V or 5V, whereas the DS1307 operates only with 5V.

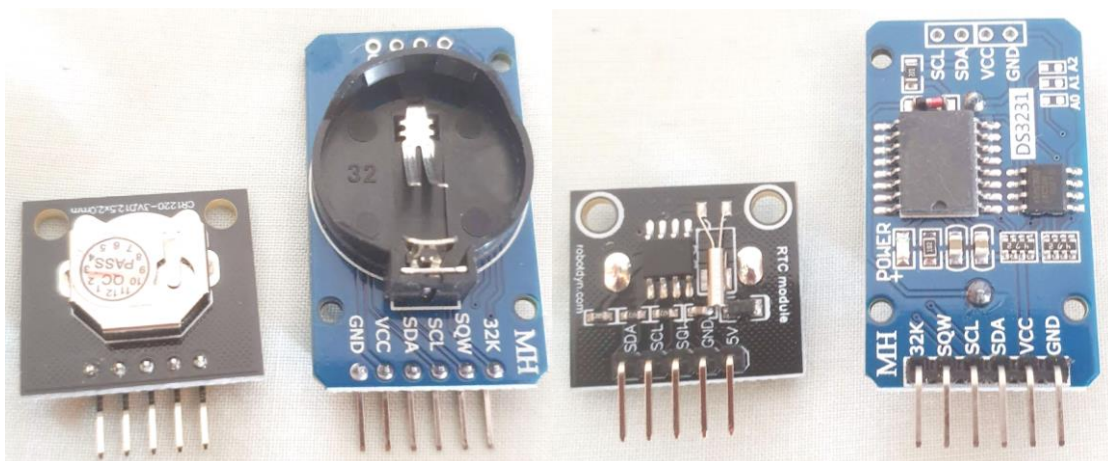


Figure 44: DS1307 and DS3231 RTC Modules

Only four wires are needed for the I2C connection. Wiring diagrams for the Arduino with DS1307, and the ESP32 with DS3231 are shown in Figure 45 and Figure 46.

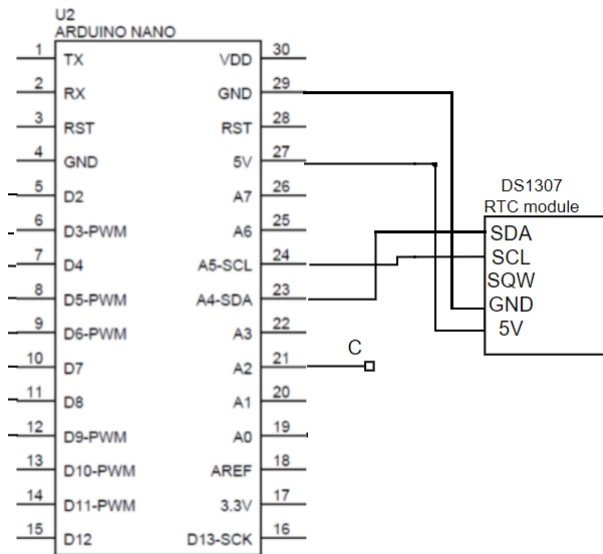


Figure 45: Wiring diagram for DS1307 and Arduino Nano

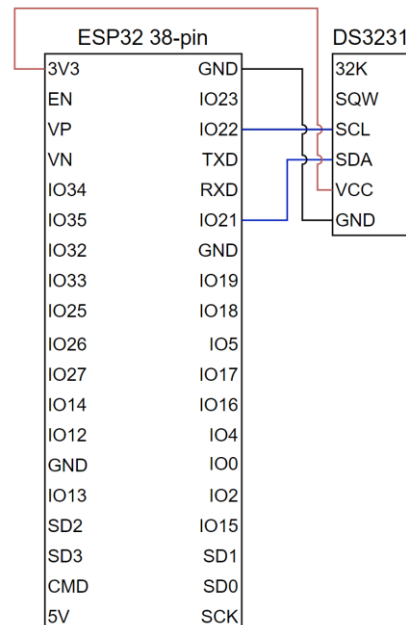


Figure 46: Wiring diagram for ESP32 and DS3231

In ESP32, I2C can be assigned to any pins, but the standard pins are SDA=21, and SCL=22. Both the DS3231 and DS1307 were successfully tested with the Arduino and ESP32 (the 5V logic of the DS1307 did not seem to damage the ESP32).

6.3.3 Future improvements

The measured RTC or NTP time could be used to set how often measurements are taken. The software is currently using a simple time delay between loops, but due to the time taken to conduct a loop, this may mean that measurements are always taken slightly after the desired sampling period. Using the true time would ensure the sample frequency is exactly as intended.

Although the RTC module is highly accurate, its accuracy could drift over time. If Wi-Fi is available, it may be better to preferentially take the time/date information using NTPClient. This was not implemented in the initial prototype code, but could easily be incorporated in the future.

7 Diverting DC Energy

The Open Energy Monitor [13] has several examples of energy diverters, using different control methods such as burst mode, phase control or pulse width modulation. These work by switching the power to the load on and off at different rates. They do this using a triac, for AC power control. In our system, we aim to divert the excess DC power from the solar panels, before the charge controllers and inverters, hence accessing more of the excess power that cannot be taken by the charge controllers.

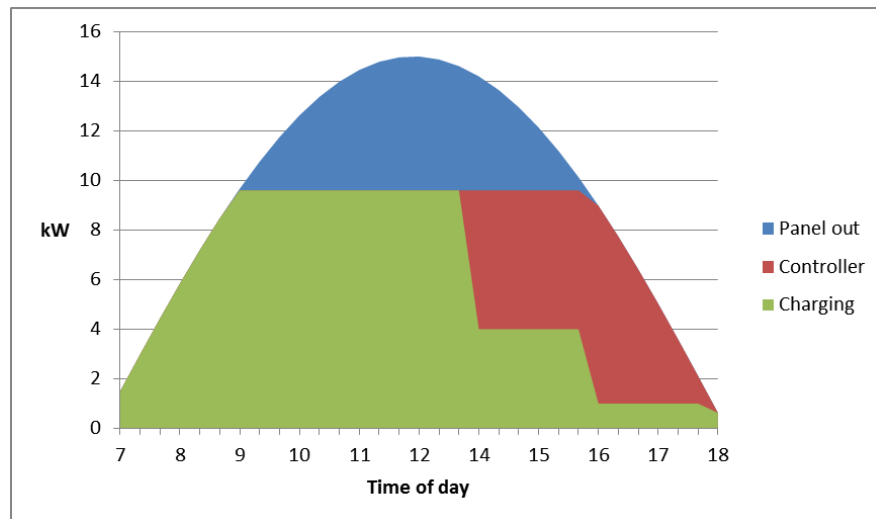


Figure 47: sample comparative system power curves to illustrate excess power capacity

For example, in the system modelled in Figure 47, the solar charge controller has a maximum power of 9.6kW. But since performance of the solar panels may be suboptimal over time, or in particular weather conditions, 15kWp of solar is installed. The charge controller uses the solar power output coloured in green to charge the batteries for system usage. Existing energy diverter technologies use the logic switching of the charge controller to harness excess solar output once the batteries are charged. But this is still subject to the power limits of the charge controller. Thus in the model here, the red area is the spare system capacity for load diversion. But the blue area – potential panel output which the controller is unable to draw – is not available to these solutions. By diverting the excess solar DC power, we are able to utilise both the blue and red areas of this model for diversion loads.

For DC power switching, a solid-state relay is more appropriate than the triacs used for AC. This takes a low voltage input (e.g. from an Arduino or ESP32 digital output pin), to control whether a higher voltage is turned on or off, and electrically isolates the two systems for safety.

The most likely use of diverted energy would be to power an immersion water heater. For a 2kW immersion heater, currents would be around 8-9A. Two solid state relays were bought from Amazon to test:

5. Heschen Single Phase DC Solid State Relay SSR-60DD 3-32 VDC/24-220VDC 60A
6. Sourcing map ASH-10DD 4-32VDC to 5-250VDC 10A Single Phase Solid State Relay DC-DC Relay

Interestingly, the relay rated for input voltages 4-32V worked when operated with a 3.3V signal from the ESP32, whereas the one rated for 3-32VDC would not switch on. The lower rated relay also had poorer reviews on Amazon, so only the 'Sourcing map ASH-10DD' was used in further testing.

After the solid-state relay had been proven to turn on and off following a simple HIGH/LOW digital write to the ESP32 controlling pin, the relay was left on with a current of 9.5A running for about 10 minutes, to test how hot it would get. The temperature rose significantly, until the DC power supply being used broke and stopped showing the current. (Note that that DC Power Supply was also bought from Amazon, and this highlights the importance of buying quality electronics from reputable sources. The power source was later replaced with a more reputable Farnell Power Supply which was able to repeat the tests with no problems). When asked, the seller said the relay should be able to operate at 10A continuous load current, provided it is bolted to a suitable heat sink.

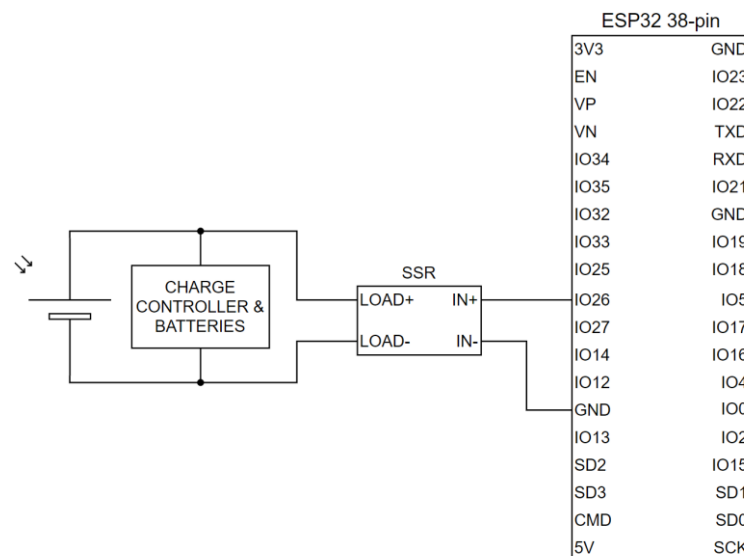


Figure 48: Solid State Relay Wiring Diagram

8 "AI" switching logic

Switching in a "solar-side" diversion load is more complex than a "battery-side" diversion load, because it isn't possible to use a logic signal from the charge controller to do so (the traditional approach uses a programmable relay in a charge controller which can be programmed to trigger when the charge controller enters "float" state, ie when the batteries are charged).

We are essentially having to balance two loads – the charge controller and the diversion load – from a single supply (the panels) whilst ensuring that the priority load (the charge controller) does not "suffer" from the diversion load being switched in.

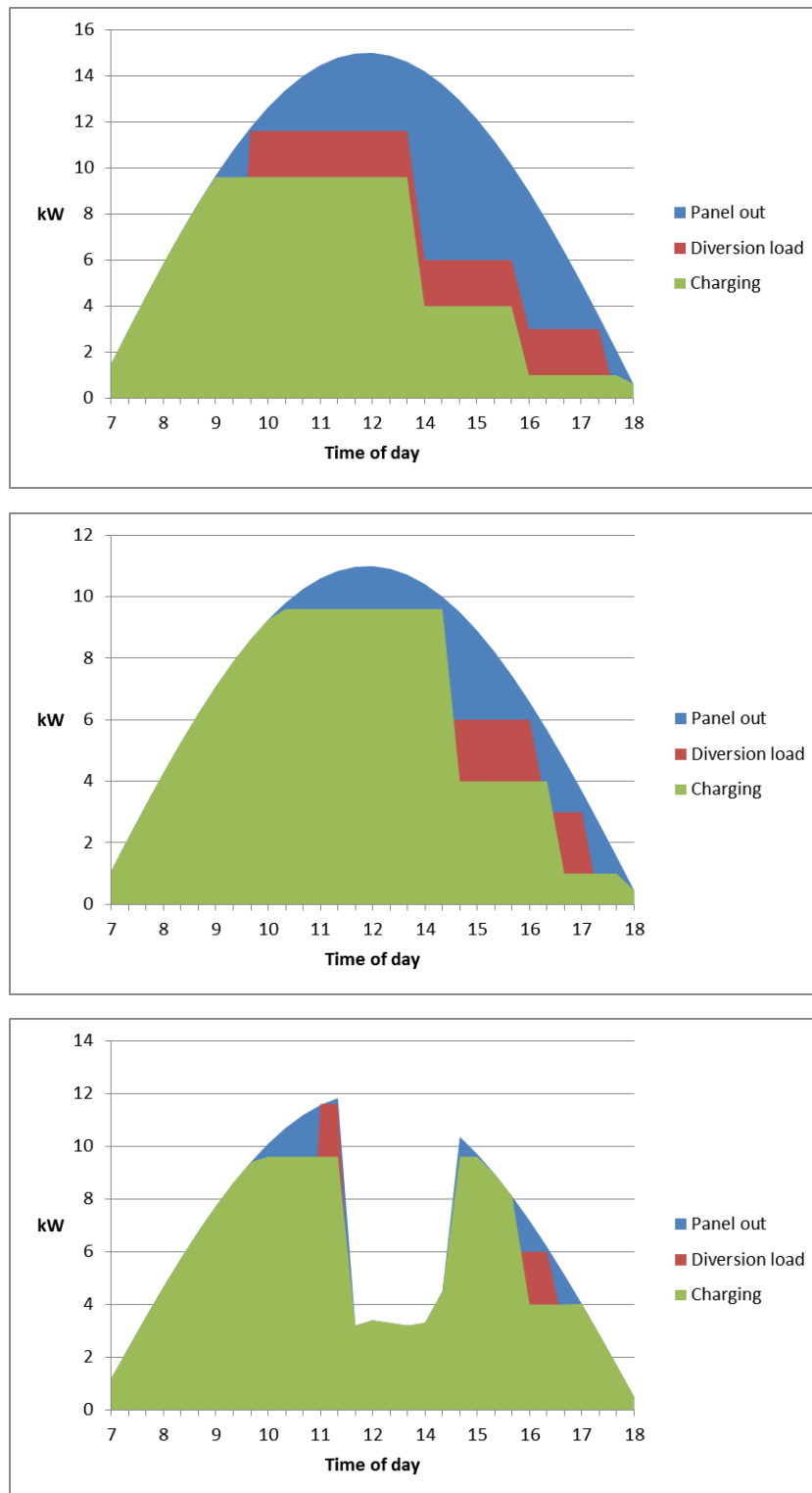


Figure 49: Different load scenarios (sunny day, hazy day, day with transient cloud)

The fundamental of switching the diversion load in is for the device to attempt this whilst continuing to monitor the current flowing to the charge controller. Assuming the operating voltage of the panel remains invariant, if the dc current flowing to the charge controller remains the same, then the panels are providing sufficient power for both loads.

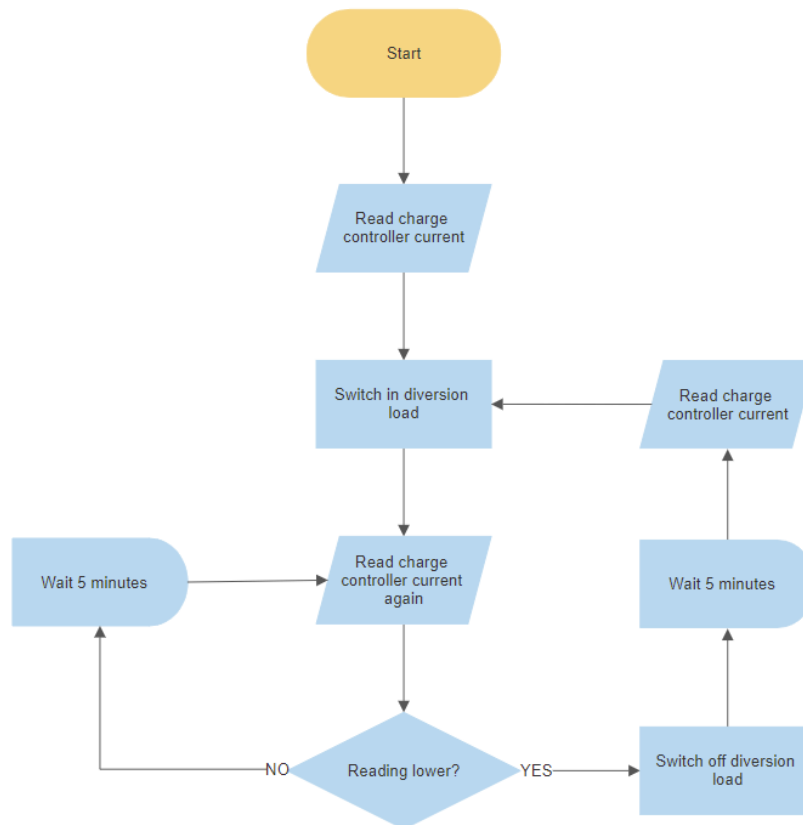


Figure 50: Basic switching algorithm for diversion load

In practice, depending on the utility of the particular diversion load being used, a greater delay than 5 minutes might be appropriate. A shortcoming of this basic approach is the repeated switching requirement, even for system states where no excess load is likely. With a combination of a real-time clock and ability to store system parameters on the SD card, the following machine learning elements are also possible:

- a) learn the maximum power draw of the priority system load (charge controller), and do not attempt the diversion load switching loop until this is exceeded each day
- b) store the time of day at which the diversion load is successfully switched on. On subsequent days do not attempt to switch the load in until 30 minutes before this time
- c) adaptive logic for delay timings – when diversion load switching is unsuccessful, increase the delay time with each iteration to reduce unnecessary cycles

More sophisticated learning and control logic is also possible for known system configurations and use cases, but this detracts from the universal application of the basic algorithms. Each of these algorithms has been coded, but we await the ability to deploy the controller in the field to be able to test their functionality.

9 Full Wiring diagram

The full wiring diagram for the first fully integrated prototype is shown in Figure 51 and Figure 52. The high voltage connections are to be wired on a separate board, helping isolate them, and remind the user/manufacture to take extra care when constructing or touching these components. Orange boxes indicate components which have an external connection, which must be considered when placing the components on the board, for easy access (e.g. an SD Card, or USB connector). It is assumed the AC current CT sensors will be plugged into the low voltage board, and can be removed during transportation.

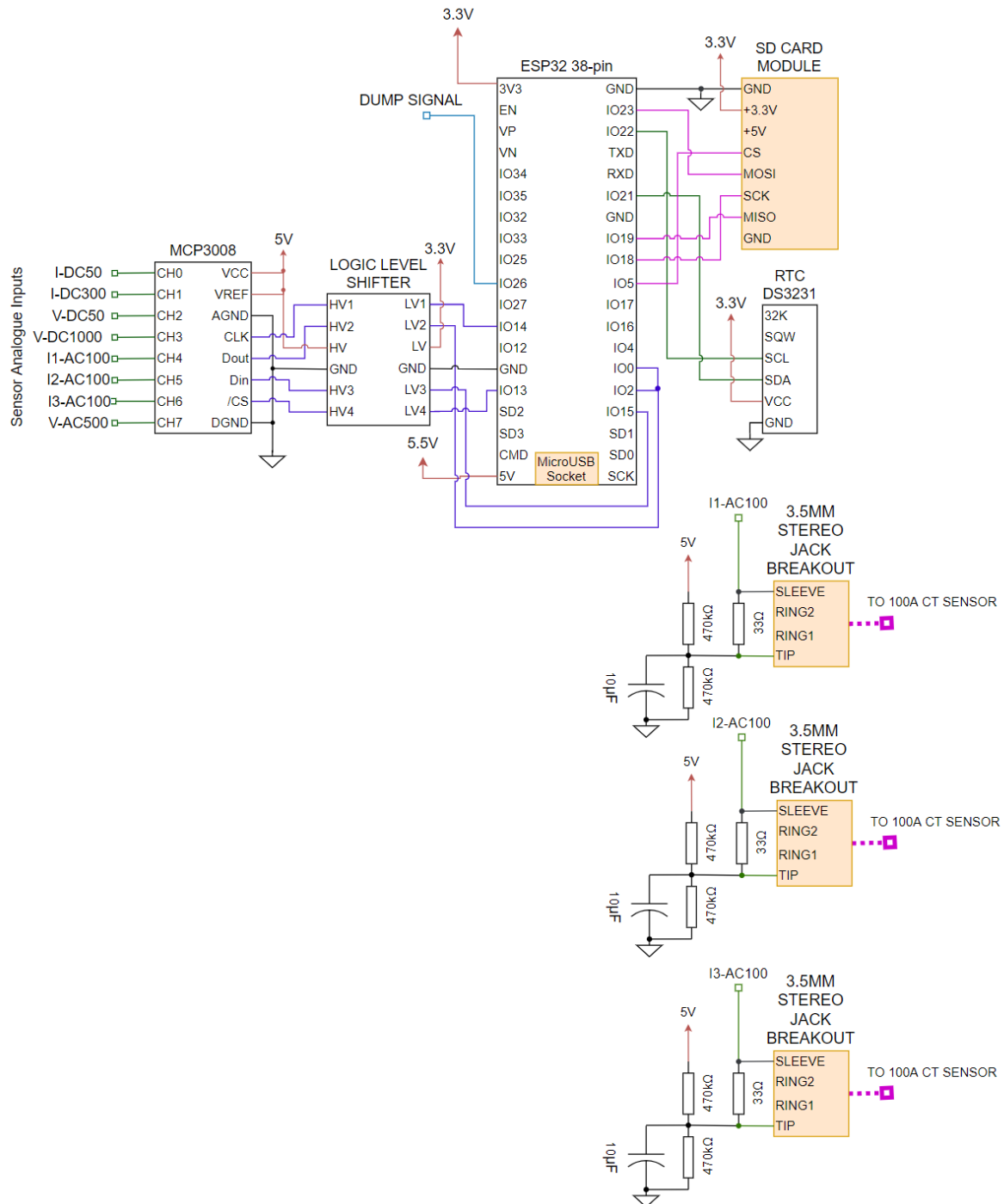


Figure 51: Wiring diagram for low voltage board. Orange boxes show components that have external connections (Updated on 20/9/2021 as CT sensor wiring was incorrect)

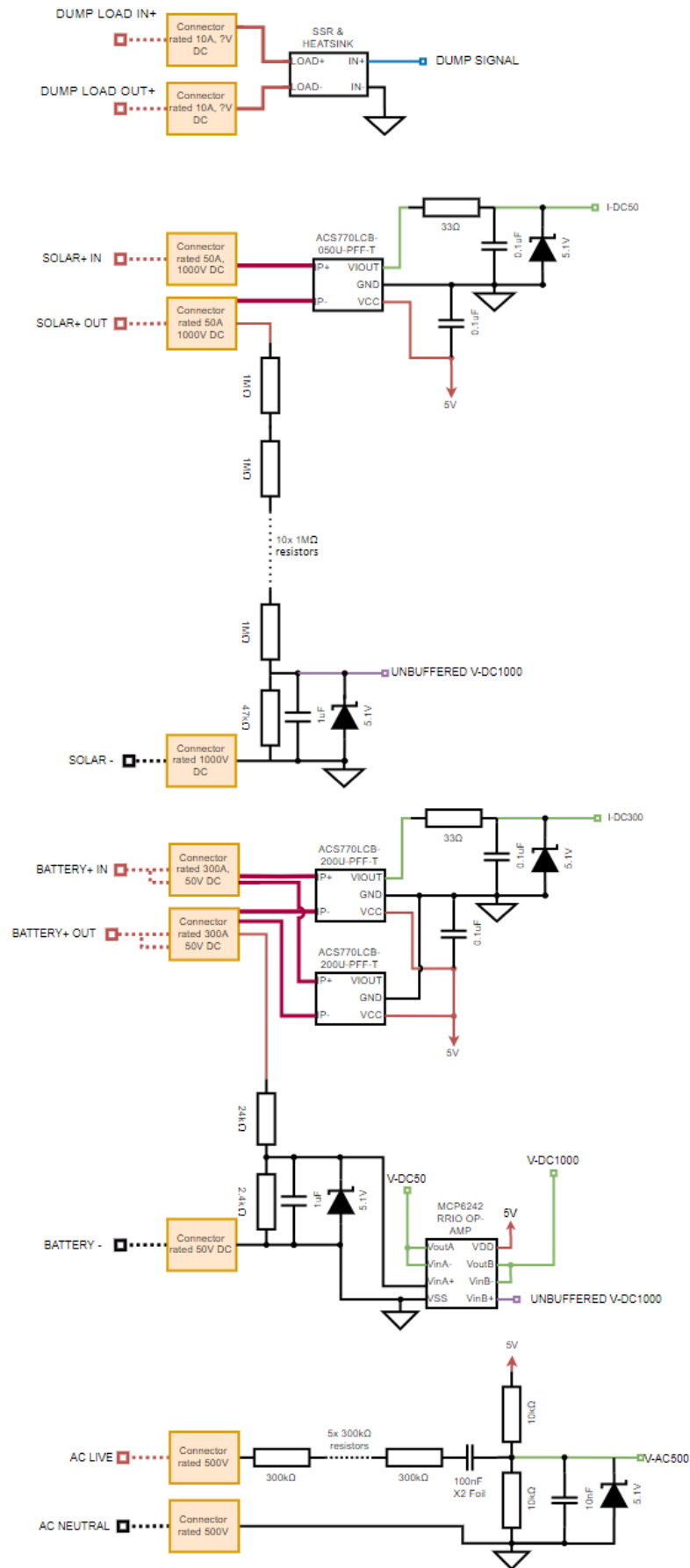


Figure 52: Wiring diagram for high voltage board, showing external connectors in orange boxes

Note that as previously mentioned, the AC and DC sensing circuits are not electrically isolated from the microcontroller. Indeed, they are all being forced to share a common DC ground and AC neutral. This is not recommended for safety reasons, and although has been shown through low voltage testing to work, more research should be undertaken to isolate these circuits if they are to be installed somewhere unsupervised for any length of time.

10 Prototyped product

10.1 Wiring

For the initial prototype, something more permanent and transportable than a breadboarded circuit board was desired, but faster and simpler to produce than a formalised printed circuit board which would need an additional design stage in specialised software.

Perforated boards were chosen and the components laid out in a manner to try and minimise the need for additional wired connections, whilst keeping a good separation between high and low voltage lines to avoid arcing. (Note that the arcing distance at 1000V in air is only 0.15mm, so a separation of two perforated holes should be more than sufficient [14]). Any components needing external connections for voltage/current sensing or power supplies etc. were placed on the outside edges. Figure 53 shows some of the layout planning that was done for the microcontroller board before any components were soldered, although this was deviated from slightly during the build process as better layouts were found. Several different permutations were tested before deciding on the final one to solder.

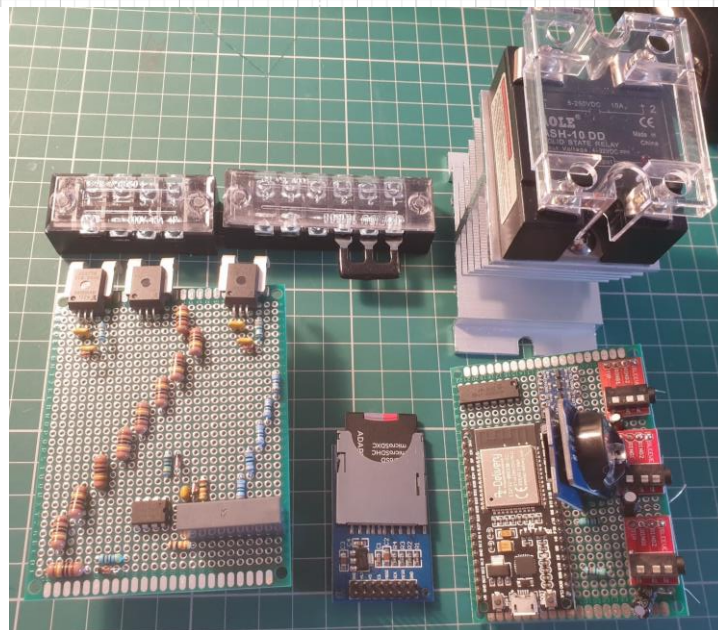
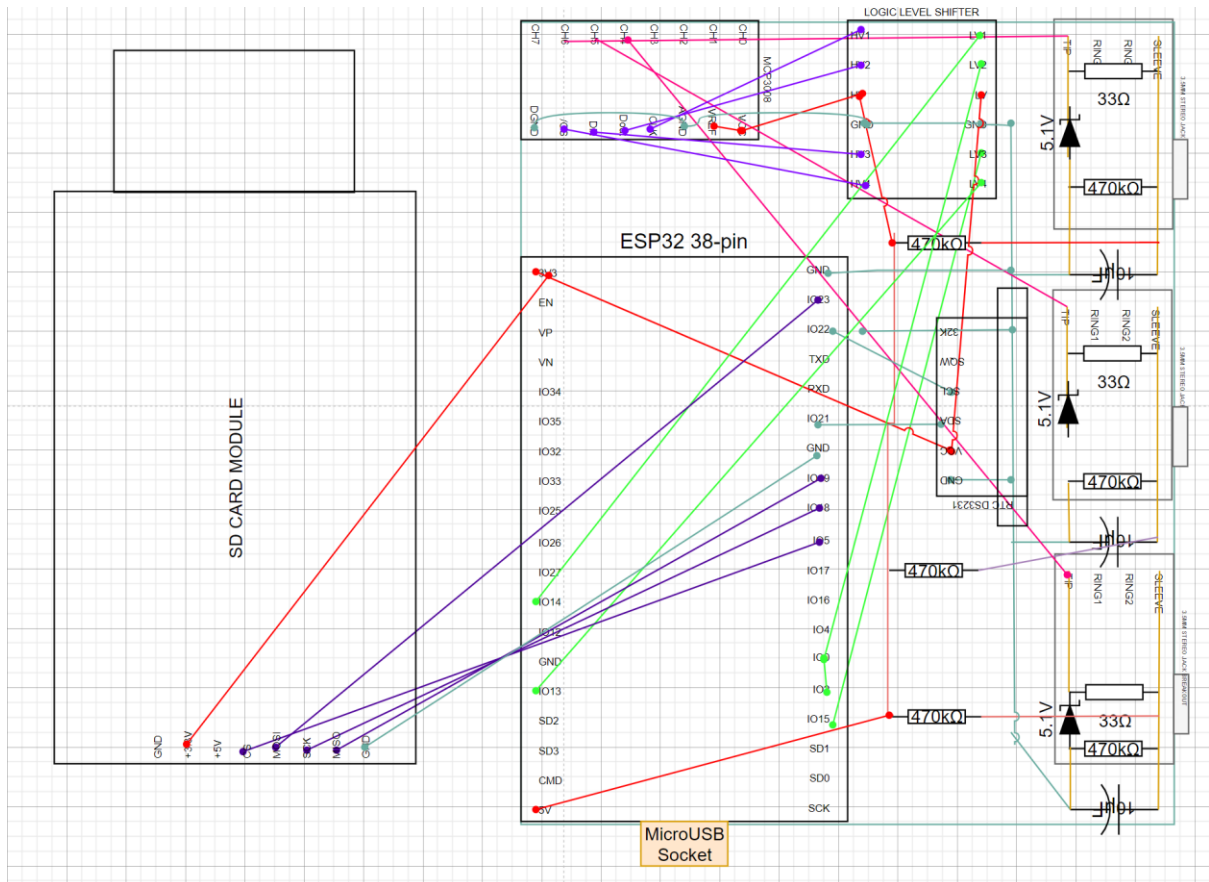


Figure 53: Planning for layout of prototyped circuit. Note that a few changes were made during the built process as it was found certain parts did not fit as easily.

Connections within a board were preferentially done using the existing component wires (e.g. the legs of a resistor, bent to reach the next component), and using single core, insulated wire where this was not possible. All soldering was done on the underside of the board. For the first board, insulated wires were attached by bending a stripped end of a wire in a loop around the legs of a component, and then soldering. Due to the thickness of the wire, the loops of adjacent connections could end up extremely close, at risk of shorting if not done carefully, as shown in Figure 54. For this

reason, later connections were done by placing the wires on the top surface of the board as opposed to the underside, feeding the stripped ends through holes, and bending the edges to just touch on the connection point to solder. This can be seen with the red and black wires in Figure 55.

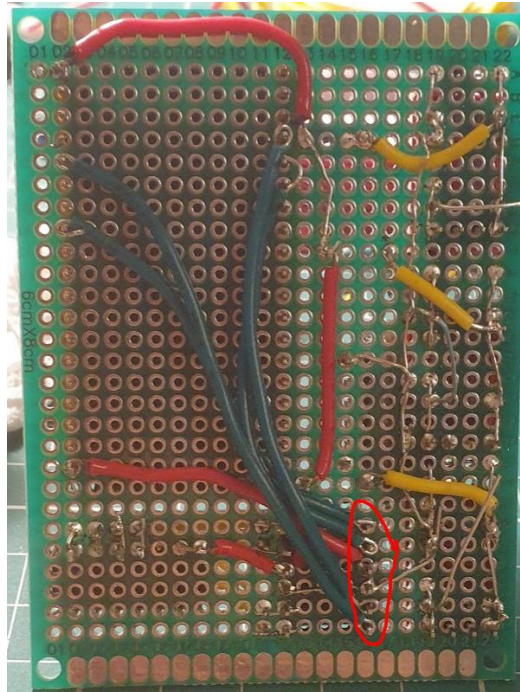


Figure 54: Underside of microcontroller protoboard during assembly process. Location of many looped wire connections in close proximity is circled in red

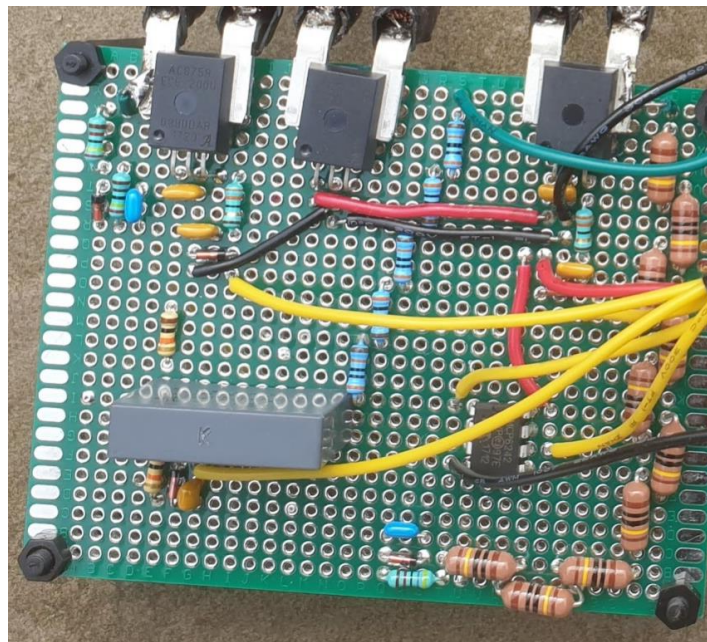


Figure 55: Top side of high voltage board

Multi-core wire was used when soldering between boards. The wire was fed into the board from above and soldered beneath for some strain relief. This can be seen in Figure 56.

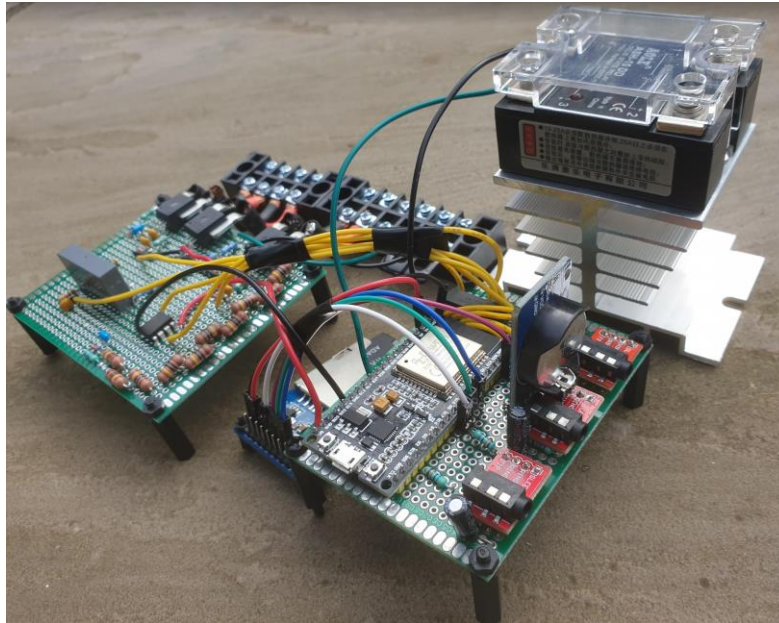


Figure 56: Fully soldered electronics

10.2 External connections

Finding terminal block connections rated for the high voltages and currents desired was found to be tricky. Due to time limitations, smaller connectors rated for only 600V 15A were used. This would enable basic testing, but would need upgrading for higher-end testing. This must be addressed in future designs.

10.3 DC current connections

As found in preliminary testing, the data and power legs of the DC current hall effect sensor are extremely sensitive to twisting forces and liable to breaking. To try and provide some strain relief, the sensors were secured to the perforated board using hot glue, before soldering. Thick wires were soldered to the high current legs of the sensors, and heat shrink applied around the outside. The thick wire was not very pliable, and made accurate placement near the connectors vital.

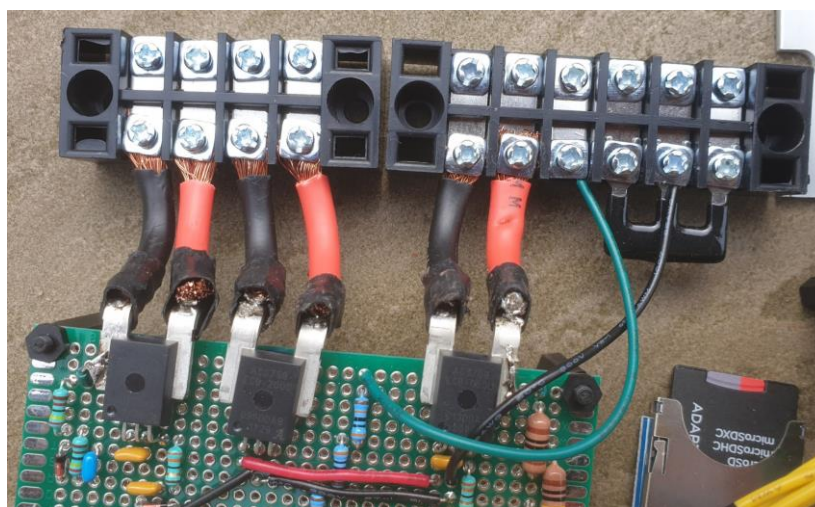


Figure 57: DC Current, DC Voltage and AC Voltage connectors

10.4 Solid State Relay

Thermal paste was applied to the heatsink surface, and the solid state relay bolted on. It is believed this heatsink should most likely be mounted in a location which allows the heat to dissipate, potentially outside of the electronics box. For the initial prototype, we expect lower currents, so have chosen to mount it inside for safety and neatness. It is believed the temperatures should not rise so far to melt the plastic enclosure.

10.5 Enclosure

A plastic junction box was bought, large enough to enclose all the components. Holes were punched out where wires were required for sensing to access the terminals, and a small hole drilled to allow the microcontroller micro-USB power supply to be accessed.

Each board was screwed into the plastic box using stand-offs, to avoid putting unnecessary pressure on the underside electrical wiring.

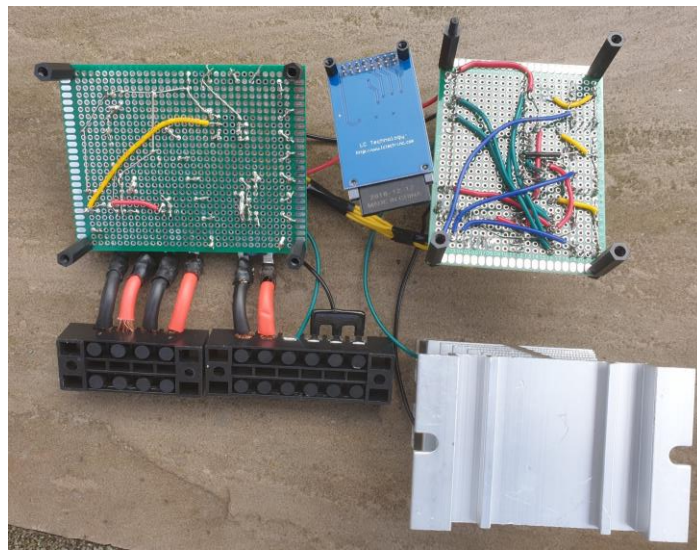


Figure 58: Underside of the soldered electronics before installing into plastic enclosure.

10.6 Final Prototype

The following photos show the fully assembled prototype, fixed via stand-offs to a plastic junction box, with holes punched out for cable exits/connections. One AC Current CT sensor is attached for illustration, although no other external connections are plugged in.

DC Current, DC Voltage and AC Voltage wires would enter the box via the closest holes and be screwed into the terminal blocks. The dump load would enter the box similarly and be screwed into the solid-state relay. The SD card can be removed and replaced while the box is open. Once in operation, the box would be shut, preventing users from touching any components and getting an electric shock.



Figure 59: Closed electronics box

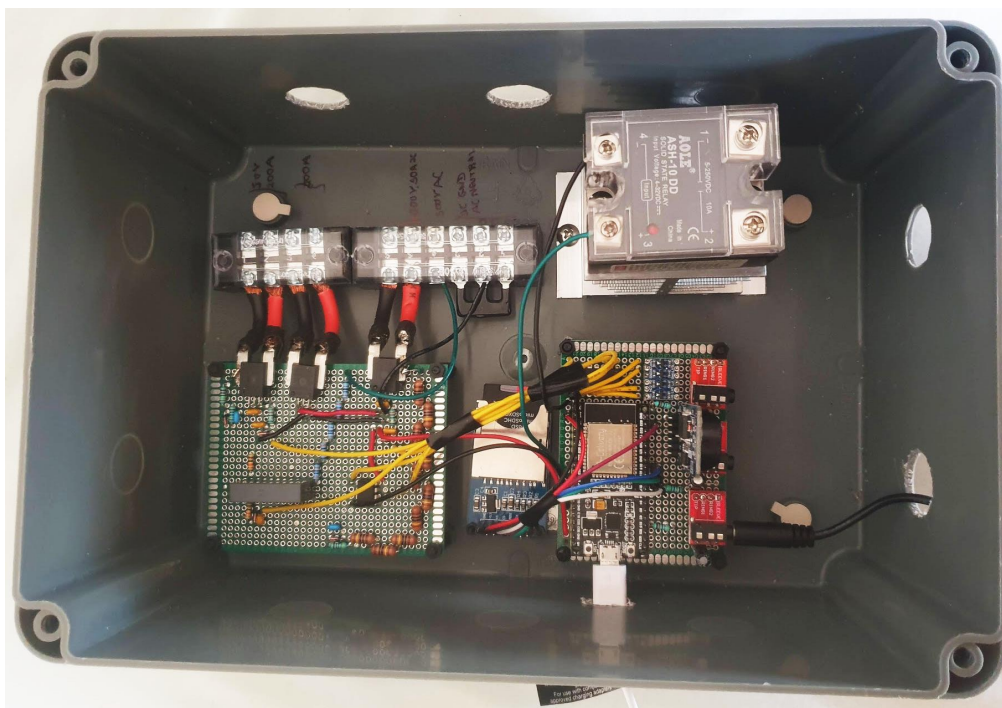


Figure 60: Top view of open electronics box

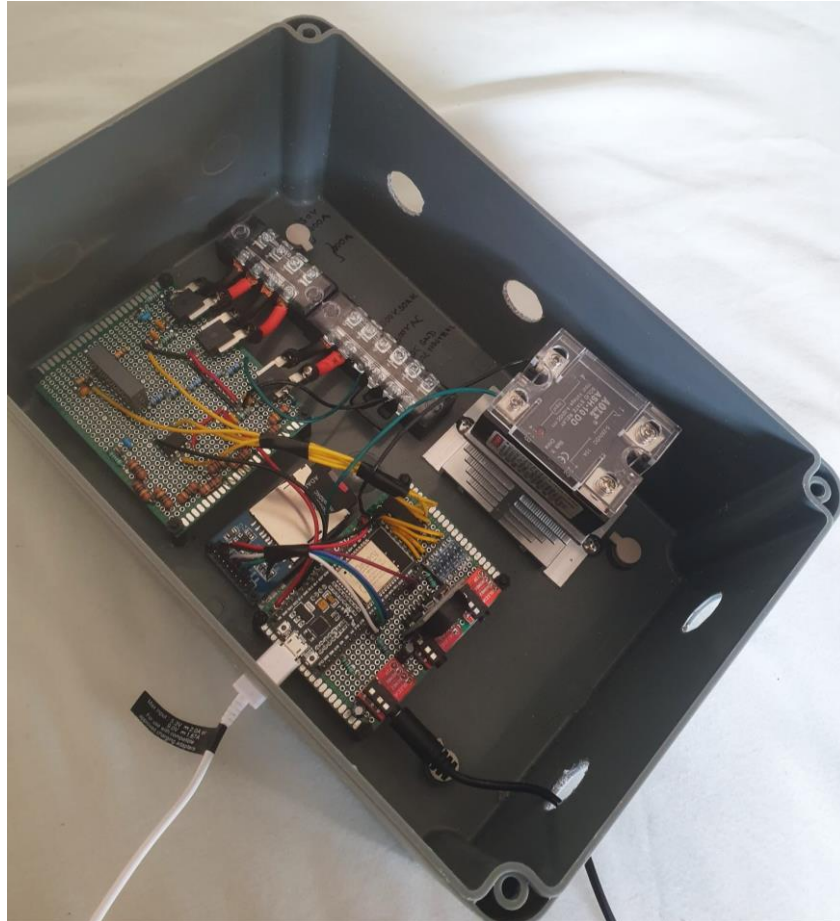
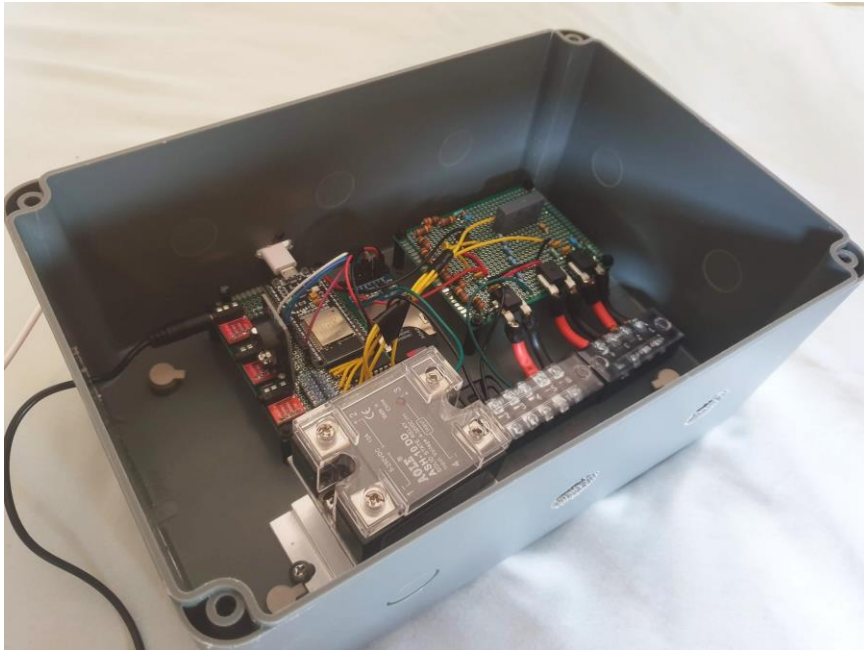


Figure 61: Side views of open electronics box, showing internal components and cable exit points

11 Prototype Testing and Evaluation

The final prototyped box was tested with a DC power supply and AC variable transformer. The performance of the final prototype is compared with original requirements table, as shown in Table 3.

Table 3: Requirements comparison table following build of first prototype

Category	Requirement	Prototype achievement
Sensing – All must work across multiple technologies	Sense 12 to 1000 VDC, Accuracy $\pm 2V$ - Two sensors, one for up to 50V, one for up to 1000V	50V sensor accurate to $\pm 0.2V$ using resistor ratio as calibration constant. 1000V sensor accurate to $\pm 1V$ (resolution is only 1.07V due to 10 bit ADC), using calibration constant increased from 214 to 219 following testing (due to resistor tolerances).
	Sense up to 300A DC, Accuracy $\pm 2A$ -Two sensors, one for up to 50A, one for up to 300A	50A sensor calibrated at 0A and 10A, and accurate to 0.2A within testing range. Testing at higher currents required. 300A sensor was not tested as only $<10A$ was available for testing which would not allow for accurate calibration, or any comment on accuracy at top end of sensor range.
	Sense 100-500V AC, Single and 3 phase (star and delta) Accuracy $\pm 2V$	Following adjustment of calibration value from 301 as used for Arduino, to 465, an accuracy of $\pm 2V$ was achieved up to 300V. Note that an X2 foil capacitor used in the AC voltage circuit is only rated up to 305V AC, so unless replaced by a higher rated circuit, this limits the safe testing voltage range.
	Sense up to 100A AC Single and 3 phase (star and delta) Accuracy $\pm 2A$	This was not tested on the final prototype due to availability of an AC current source, but earlier tests with the same sensors showed an accuracy of $\pm 1.5A$ at the lower end of the range. The sensors are rated for 100A so should work effectively once calibrated.
Data logging	Take accurate time-stamp for each measurement	Achieved using RTC Module, or alternatively, can be programmed to use NTPClient.
	Sample measurements at frequencies	Achieved using delay(x) between

	from 1s to 1hr	each loop cycle of reading and writing data.
	Write measurements to local storage device	Data is stored on local SD card
	Communicate over internet to write data to cloud	Data is communicated to ThingSpeak where it is plotted and stored
Power diversion	Programmable to apply a 5V switching signal to a solid-state relay to operate a dump load	Solid State relay is inside electronics box and can be wired to productive load.
Cost	Components cost less than £100 total	<p>Prototype cost around £115 to manufacture, but this is anticipated to drop to £75 if components were purchased in bulk.</p> <p>Note that this price is for the current design, and so does not include appropriately sized/rated connectors and wiring, and future designs may also require additional circuit breakers/fuses and isolating circuits, increasing the cost. See Section 14.2 for detailed BOM and costings</p>
Installation	Can run from a 5V DC supply	All components take their power from the ESP32, which uses a 5V Micro-USB as its power source
	External connectors, for easy installation in any location without soldering.	<p>The chosen connectors are rated at 300V, 15A, so despite the electronics meeting the sensing requirements, the higher voltages and currents cannot be applied.</p> <p>The connectors do however not require soldering, and do allow easy installation.</p>
Safety	All electronics enclosed within an insulated or grounded box, so there is no risk of electrocution if outside is touched during operation	All electronics are enclosed within an ABS junction box with small holes for wire egress and ingress.
	Contain appropriate fuses/diodes to prevent overloading microcontroller or other electronic components	5.1V Zener diodes were placed across the ADC inputs to prevent damage if a measurement is unexpectedly high.

12 Future Improvements Summary

Throughout this report, a number of potential, and necessary improvements have been highlighted. These are summarised in this section.

12.1 Electronic hardware improvements:

12.1.1 Urgent- necessary for safety reasons, or to meet the initial technical requirements

1. Replace the 8 channel, 10 bit ADC chip (MCP3008) with a chip with at least 10 channels, to enable simultaneous sampling of all 3 AC voltages, 3 AC currents, 2 DC currents and 2 DC voltages.
 - The resolution should be 10 bit or above to match the MCP3008, and would require rewriting the Arduino MCP3008 library that is currently being used. One example chip found is the Texas Instruments, 11 channel, 10-bit ADC chip, TLC1543IN. More research is needed to check whether the sampling frequency is sufficiently high for AC measurements.
2. Isolate the high input DC and AC voltages from the microcontroller for improved safety (currently sensed using potential dividers).
 - One method might be to use optocouplers
3. Conduct an assessment on potential failings of electrical components, and add circuit breakers at appropriate locations for 'above rated' currents.
4. Check ratings required of X2 foil capacitor in AC voltage circuit, and find a more appropriately rated capacitor that can go to 500VAC, not just 305VAC.

12.1.2 Highly desirable – to improve functionality

5. Understand and fix the SD card software such that it will continue to read and write after momentary power loss
 - All subsequent write attempts fail after a short power loss at present, unless the code is restarted.
6. Do more research into the adapted EmonLib Library which uses a callback function with an external ADC, to understand why it gives a calculated AC voltage lower than expected.
 - Without the adapted callback code, feeding the result directly into an Arduino with the original EmonLib code gives the expected calculated AC voltage, so it is definitely a software issue.

12.1.3 Non critical

7. Optimise the values of decoupling capacitors used in all circuits, and of the noise management resistor and capacitor in the DC current sensing circuit.
 - More research is needed to understand what frequency ranges potential noise lies in, to determine what values of capacitors and resistors to use.
8. Improve software code to preferentially get time from NTPClient over RTC if WiFi is available, and update the RTC time.
 - This could be more accurate as the RTC module accuracy may drift over time)
9. Improve software code to take readings based on a timing interrupt, rather than a delay, to ensure that the time interval between readings is a whole number of seconds.
 - The current delay method means that the time between samples is equal to the delay, plus the time taken to conduct the loop (take measurements, make calculations, and write data). Using the true time would ensure the sample frequency is exactly as intended
10. Enable Deep Sleep of ESP32 between readings - useful for power saving
11. Work/Verify out the voltage phase shift introduced by the sensing software, to enable accurate measurement of apparent and true power, and save this data along with sensor data.
12. Explore using an alternative microcontroller which uses data connectivity instead of Wi-Fi.

13. Improve the method used for writing to the cloud

- Currently using ThingSpeak, but the graphical interface lacks customisability.
- We could write our own software to receive and graph the results, or explore in more detail Home-assistant.io and Emoncms.org

12.2 Practical recommendations for future improvements:

12.2.1 Urgent- necessary for safety reasons, or to meet the initial technical requirements

1. Find and install appropriately rated connectors for voltage and current sensing.
 - The current system is not rated for high enough currents. Extremely thick wire is required for currents up to 300A. More research is needed into the exact wire diameters, and how these would be connected (most likely with a bolt and lug, or MC4 connectors).
 - The high currents may need to be split into two prior to entering the box (we split the high currents into two parallel sensors).
 - Our experience with MC4 connectors show that they can require significant force to connect and disconnect, but this may be a safer option than screw terminal blocks.
 - High rated DIN rail connectors are more readily available, but thought is needed on how to connect wires inside the box before they reach the DIN rail connectors, and how to ensure they would be an appropriate length.
 - Soldering to the DC current sensor legs was quite difficult, and when thicker wires are used, this method may be inappropriate. More research is needed into how thicker wires can be directly soldered, if lugs can be soldered directly to the sensor legs for connection to thicker wires, or if alternative better options exist.
2. Improve on solid state relay installation.
 - Do more research into how the SSR should be installed with a heatsink and whether it needs to be external to the plastic enclosure for sufficient air flow.
 - More reputable SSR sources than Amazon may be desired, as it has been found that some of their cheap electronics are not entirely trustworthy or meet the standards claimed.

12.2.2 Highly desirable – to improve functionality

3. Implement circuit in software to enable design of printed circuit board.
 - This would make assembling the circuit much quicker and easier if more prototypes are to be made.

12.2.3 Non critical

4. Add cable glands at junction box exit points
 - To prevent wildlife and insects from entering the box during operation (we have found this to be an issue in previous solar installations).
5. Solder IC dip sockets to the circuit board instead of directly soldering ICs
 - This should be done in future to reduce the risk of damaging the IC during assembly, and makes it easily replaceable if it breaks during use.
6. Use headers between boards, with flexible cable which can be plugged in.
 - For ease of prototyping, connecting cables were soldered directly to the protoboard at the location of the sensing circuit (i.e. voltage data cable soldered next to voltage potential divider). This made it easy to remember and see visually which cables linked to which circuits during troubleshooting, but practically is less suitable, as it

was found these soldered connections could easily break if the two connecting boards were moved relative to each other during assembly.

- It would be better to bring all external data and power line connections to one side of the board, and install header pins, or a socket, which can be plugged into the next board (ensuring the order of wires are clear).

13 Conclusion

Our work so far has shown us that an offgrid multi-energy reporting system, capable of interfacing with any solar minigrid, and diverting excess solar energy prior to any charge controllers, is possible at an affordable price. We believe this key functionality is worth pursuing further, and are grateful to Innovate for the opportunity to advance our skills so far, up to a functioning prototype.

We intend to take the current prototype with us when we are next able to access our sites in Tanzania, to run further tests with higher power levels and to fully test the dump load control functions.

We will be seeking to develop the product further, improving on the safety and performance in a Mk II prototype, which we plan to leave on-site to generate a long-term dataset for us. From this, we can then develop a final design and user interface which we can then accredit against relevant safety and consumer standards, so that we have a market-ready product that minigrid providers worldwide can install alongside their existing systems for additional energy saving and efficiency. We may need to seek additional funding to help us complete these further development tasks.

14 Appendix

14.1 Resistor Sizing Calculations

14.1.1 1000V DC resistor calculations

- Need a 1000V:5V divider for 5V ADC reference voltage
- Implement as 47kΩ:10MΩ
 - Use 10 x 1MΩ resistors in series to reduce power dissipation in each, instead of a single 10MΩ resistor
 - Note that ideal lower resistor value is 50.2kΩ to maximise the full range of the 5V ADC. We round down the resistor value to the closest standard resistor value (47kΩ), to avoid overloading the ADC from rounding up. This means that voltages of 1000V will show as 4.68V at the ADC.
- Maximum current draw will be $1000V / (10M\Omega + 47k\Omega) = 0.1mA$
- Total power dissipation = $0.1mA * 1000V = 0.1W$
- Voltage drop across each 1MΩ resistor = $0.1mA * 1M\Omega = 99.5V$
- Power dissipated in each 1MΩ resistor = $0.1mA * 99.5V = 0.01W$

14.1.2 50V DC resistor calculations

- Need a 50V:5V divider for 5V ADC reference voltage
- Target 0.1W power loss:
 - Max current = $0.1W / 50V = 0.002A$
 - Total resistance = $50V / 0.002A = 25k\Omega$

- Closest standard resistor combination (rounding up both values): 2.4kΩ:24kΩ
 - Results in slightly less power loss: 0.095W
 - The voltage drop and power dissipated in each resistor will be far below their rated values due to the lower voltages.

14.1.3 AC Potential Divider calculations

14.1.3.1 Test set-up: using available resistors

- If measuring 500VAC, the peak voltage will be 707V. Maximum peak to peak voltage is 1414V.
- If we let $R_1=R_2=10\text{k}\Omega$ (effective parallel resistance = $5\text{k}\Omega$), then for a 1414V:5V ratio, $R_3 = 1.401\text{M}\Omega$
- Round up to the nearest standard resistor value after splitting into eight resistors in series to spread the voltage drop and power dissipation: $200\text{k}\Omega \times 4 + 360\text{k}\Omega \times 2$
- Max voltage drop across $200\text{k}\Omega$ resistor = $1414\text{V} \times 200\text{k}\Omega / ((200\text{k}\Omega \times 4 + 360\text{k}\Omega \times 2) + 5\text{k}\Omega) = 185\text{V}$
- Max voltage drop across $360\text{k}\Omega$ resistor = $1414\text{V} \times 360\text{k}\Omega / ((200\text{k}\Omega \times 4 + 360\text{k}\Omega \times 2) + 5\text{k}\Omega) = 333.8\text{V}$
- Max instantaneous power dissipated in each $200\text{k}\Omega$ resistor = $V^2/R = 185^2 / 200\text{k}\Omega = 0.172\text{W}$
- Max instantaneous power dissipated in each $360\text{k}\Omega$ resistor = $V^2/R = 333.8^2 / 360\text{k}\Omega = 0.31\text{W}$
- Max instantaneous current = $P/V = 0.31\text{W}/333.8\text{V} = 0.9\text{mA}$
- Max total instantaneous power dissipated = $(0.9\text{mA})^2 \times (200\text{k}\Omega \times 4 + 360\text{k}\Omega \times 2 + 5\text{k}\Omega) = 1.31\text{W}$
 - These values can be adjusted if less power dissipation is required, by increasing the sizes of all resistors, keeping the same 1414V:5V ratio.

14.1.3.2 Improved set-up for prototype, using fewer resistor types and fewer number to reduce cost:

- If measuring 500VAC, the peak voltage will be 707V. Maximum peak to peak voltage is 1414V.
- If we let $R_1=R_2=10\text{k}\Omega$ (effective parallel resistance = $5\text{k}\Omega$), then for a 1414V:5V ratio, $R_3 = 1.401\text{M}\Omega$
- Round up to the nearest standard resistor value after splitting into five resistors in series to spread the voltage drop and power dissipation: $300\text{k}\Omega \times 5$
- Max voltage drop across $200\text{k}\Omega$ resistor = $1414\text{V} \times 300\text{k}\Omega / ((300\text{k}\Omega \times 5) + 5\text{k}\Omega) = 281.9\text{V}$
- Max instantaneous power dissipated in each resistor = $V^2/R = 281.9^2 / 300\text{k}\Omega = 0.26\text{W}$
- Max instantaneous current = $P/V = 0.26\text{W}/281.9\text{V} = 0.9\text{mA}$
- Max total instantaneous power dissipated = $(0.9\text{mA})^2 \times (300\text{k}\Omega \times 5 + 5\text{k}\Omega) = 1.27\text{W}$
 - Note that the area under one hump (from 0 to π) of a \sin^2 curve of amplitude 1 is $\pi/2$. If 1.27W is the equivalent of the peak power being continuous, then the approximate average continuous total power dissipated is $(1.27/\pi) \times (\pi/2) = 0.6\text{W}$
- These values can be adjusted if less power dissipation is required, by increasing the sizes of all resistors, keeping the same 1414V:5V ratio.

14.1.4 AC 100A Current Transformer Burden Resistor

- Max current = 100A
 - Primary peak current = $\text{RMS current} \times \sqrt{2} = 100\text{A} \times 1.414 = 141.4\text{A}$

- Peak current in secondary coil = Primary peak-current / no. of turns = $141.4 \text{ A} / 2000 = 0.0707 \text{ A}$
- To maximise measurement resolution, voltage across the burden resistor at peak-current should be $\frac{1}{2}$ the analog reference voltage. Let $V_{\text{ref}}=5\text{V}$.
 - Ideal burden resistance = $(A_{\text{REF}}/2) / \text{Secondary peak-current} = 2.5 \text{ V} / 0.0707 \text{ A} = 35.4 \Omega$
 - Round down to closest standard resistor value (to ensure V_{ref} is not exceeded at ADC) = 33Ω
- Calibration value = current for 1V output = Primary current / output voltage = $100 \text{ A} / ((100 \text{ A} / 2000 \text{ turns}) * 33 \Omega) = 60.6$

14.2 BOM

A rough BOM for the current design is detailed in Table 4. This brings the total cost per device to around £115 if manufactured as a one-off, and around £75 if components like resistors are bought in bulk (making the price per unit about 1/10th the cost).

Note that this price is for the current design, and so does not include appropriately sized/rated connectors and wiring, and future designs may also require additional circuit breakers/fuses and isolating circuits, increasing the cost.

Table 4: BOM for first prototype design

Category	Component	Purpose	Example Purchasing source	Requirements of specific component	Quantity	Total cost (incl. VAT)	Estimated total cost if bulk
Resistors	1M Ω through-hole resistor	1000V DC Voltage divider (x10)	RS 830-9217	max ratings >99.5V, 0.01W	10	2.3	0.2
	47k Ω through-hole resistor	1000V DC Voltage divider (x1)	RS 707-8369	max ratings >5V, >0.5mW	1	0.029	0.003
	24k Ω through hole resistor	50V DC Voltage Divider (x1)	RS 148-821	Max ratings >50W, 0.09W	1	0.05	0.005
	2.4k Ω through hole resistor	50V DC Voltage Divider (x1)	RS 148-590	Max ratings >5W, 0.01W	1	0.146	0.015
	33 Ω through hole resistor	DC current sensing circuits (x2), AC current sensing circuits (X3)	RS 148-130		5	0.1	0.01
	10k Ω through hole resistor	AC voltage circuit (x2)	RS 131-378		2	0.09	0.009
	470k Ω through hole resistor	AC current circuits (x6)	RS 149-149		6	0.5	0.05

	300Ω through hole resistor	AC voltage circuit (x5)	RS 683-4216	max ratings >185V	5	0.1	0.01
Capactors & diodes	0.1uF capacitor	DC current sensing circuits (x2)	RS 699-5002		2	0.3	0.03
	1uF capacitor	DC voltage potential dividers (x2)	RS 811-8364		2	0.8	0.08
	10uF capacitors	AC current circuits (x3)	RS 747-1913		3	0.8	0.08
	100nF X2 foil capacitor	AC voltage circuit (x1)	RS 165-0206	>=500V AC (note this capacitor does meet the requirement)	1	0.6	0.06
	10nF capacitor	AC voltage circuit (x1)	RS 841-0733		1	0.4	0.04
	5.1V Zener Diode	1000V and 50V DC Voltage dividers (x2), AC CT sensor circuits (x3), DC current sensor circuit (x2), AC voltage circuit (x1)	RS 544-3597		8	0.848	0.08
Sensors, ICS & Microcontrollers	MCP6242 RRIO OP-Amp	DC voltage potential dividers	RS 403-165		1	0.36	0.036
	MCP3008 ADC chip	External ADC chip.		>=10bit, >=8 channels	1	2	0.2
	Logic Level Shifter	From ADC to microcontroller		>=4 channels	1	1.95	1
	ESP32 dev module	Microcontroller	Amazon/ebay		1	6	5
	DS3231 RTC Module	For keeping time when power lost	Amazon/ebay	3.3V logic	1	6.5	5
	SD Card Reader Module	For storing data locally	Amazon/ebay	3.3V logic	1	2.3	1
	ACS758LCB-200U-PFF-T	300A DC current sensor (x2)	RS 170-9119	Max ratings >=150A	2	11.4	8.2
	ACS758LCB-050U-PFF-T	50A DC current sensor (x1)	RS 170-9121	Max ratings >=50A	1	6.29	4.1
	100A CT Sensor	AC Current circuit (x3)			3	13	10
Switching circuits	SSR	Operating dump load	Amazon/ebay		1	20	20
	Heatsink	Operating dump load	Amazon/ebay		1	10	10

	heatsink thermal paste	Operating dump load	Amazon/ebay			0.5	0.5
Connectors, boards and enclosures	3.5 Stereo Jack breakout board	AC current circuits (x3)	Amazon/ebay		3	3	1
	Standoffs	For each circuit board	Amazon/ebay			0.1	0.01
	Terminal Block 600V 15A	3 Ground connections, 7 sensor lines	Amazon/ebay			5	3
	Single and multistrand wire	Connections between and on boards	Amazon/ebay			0.5	0
	Perforated/printed circuit board		Amazon/ebay			0.5	0.5
	Screws & nuts		Amazon/ebay			0.3	0.03
	Plastic enclosure		Amazon/ebay			15	2
	Heatshrink		Amazon/ebay			0.1	0.01

14.3 Code

Full code used for testing is available at <https://github.com/nasherxh/EnergyMonitor2> , including individual tests for separate sensors, and functionalities.

The final integrated test is copied below for reference. Note that the AI dump load controller logic has not been included here as although the code was written, it has not yet been tested on live solar data. The act of applying a switching signal to a solid-state relay has however been tested, and combining the monitoring side with the dump load controller should be straightforward.

```
/******
ESP32
Connect to MCP3008 ADC chip via HSPI
Connect to SD card via VSPI (IO23=MOSI, IO18=SCK,IO19=MISO,IO5=CS)
Connect to Wifi to upload data to ThingSpeak Channel -->
    Note that ThingSpeak has only 8 channels, and there is a rate limit of 1 reading every 15 seconds.
Write data from MCP3008 to SD card.
Get time from RTC Module

Uses forked EmonLib library with callback method, enabling use of external adc
    Library fork available here: https://github.com/PaulWieland/EmonLib/tree/4a965d87061c19ad8b0a35bb173caada014ecbd9

Remember to tie GPIO2 to GPIO0 if using for HSPI (to allow flashing ESP32)

Note that if power is lost momentarily from SD card, it will fail for all subsequent appends
--> Need to find a way of restarting connection each time such that this is not an issue
(I think it is software-fixable as restarting the program fixes the problem)

*****/
// Libraries for SD card
#include "FS.h"
#include "SD.h"
#include <SPI.h>
// Define CS pin for the SD card module
#define SD_CS 5
// String for writing data to SD card and serial port
String dataMessage;

// Libraries for WiFi
#include <WiFi.h>
/***/ Replace with your network credentials *****/
const char* ssid = "VM4995142";
const char* password = "t3smWnchwrvx";

//Variables to save date and time for RTC
char dateStamp[] = "YYYY-MM-DD";
char timeStamp[] = "hh:mm:ss";

//Libraries for RTC Module
#include "RTCLib.h"
RTC_DS3231 rtc;

// Save reading number on ESP32 RTC memory
RTC_DATA_ATTR int readingID = 0;

//Library for MCP3008 ADC chip
#include <mcp3008.h>
//initiate an instance of MCP3008
mcp3008 mcp = mcp3008();
//Setup pins for ADC SPI interface. Example shows HSPI options*****
const uint8_t _SS = 13;//15;
const uint8_t _MISO = 2;//12;//Note that if using '2', must tie pins 2 and 0 together. If using 12, remove connection during code upload.
const uint8_t _MOSI = 15;//13;
const uint8_t _SCLK = 14;

//Calculated Sensor Values (shown below in order of ADC pin assignment)
float I_DC50;
float I_DC300;
float V_DC50;
float V_DC1000;
float I_AC100;
float I2_AC100;
float I3_AC100;
float V_AC500;

//Calculated Power Values **note this information is not currently sent to thingspeak or recorded on the SD Card*****
//for future interest*****
float RealACPower1;
float RealACPower2;
float RealACPower3;
float ApparentACPower1;
float ApparentACPower2;
float ApparentACPower3;
float ACPowerFactor1;
float ACPowerFactor2;
float ACPowerFactor3;

//ThingSpeak server settings
//REPLACE WITH YOUR THINGSPEAK CHANNEL NUMBER AND API KEY IF DIFFERENT*****
#include <ThingSpeak.h>
unsigned long myChannelNumber = 1139755;
const char * myWriteAPIKey = "F9Q1ROVIZW8TLDYE";
WiFiClient client;

//Library for AC Voltage and Current calculations
//note this is not the standard EmonLib as it has been adapted with a callback instead of directly reading an analog pin
#include "EmonLibADC.h"

//Create instance of EmonLib EnergyMonitor Class for each phase
EnergyMonitor emon1;
EnergyMonitor emon2;
EnergyMonitor emon3;
const int V_AC500_Pin = 7;
const int V_AC500_Calibration = 465;//301; ****May need tuning (Should be sensor voltage when have 1V on ADC)*****
const int V_AC500_Phase = 0;//1.7; ****May need tuning. Is phase shift due to sensor*****
const int I1_AC100_Pin = 4;
const int I2_AC100_Pin = 5;
const int I3_AC100_Pin = 6;
const float I_AC100_Calibration = 60.6; //*****May need tuning (Should be sensor current when have 1V on ADC)*****
```

```

//Calibration/set-up values for DC Voltage
const int V_DC50_Pin = 2; //ADC Pin
const int V_DC1000_Pin = 3; //ADC Pin
const int Vref = 5; //reference voltage of adc
const int adcLevels = 1024; //No of possible levels for adc. In this case is 10 bit adc MCP3008 = 1024 levels

//****Calibration values may need tuning (should be voltage when have 1V on ADC)*****
const int V_DC50_Calibration = 11;
const int V_DC1000_Calibration = 219;

//Calibration/set-up values for DC Current
const int I_DC50_Pin = 0; //ADC Pin
const int I_DC300_Pin = 1; //ADC Pin

//*****Raw values input for calculating calibration constants m & x. May need tuning *****
int I_DC50_RawAt0A = 122; //ADC reading at 0A
int I_DC50_RawAt10A = 241; //ADC reading at 10A
const float m1= 10.0/(I_DC50_RawAt10A - I_DC50_RawAt0A); //gradient of mapping adc output (x) to current (y)
const float c1= -m1*I_DC50_RawAt0A; // y-intercept for graph of current (y) vs adc output (x)

int I_DC300_RawAt0A = 122; //ADC reading at 0A
int I_DC300_RawAt10A = 137; //ADC reading at 10A ****note this hasn't been tested yet, and should be calibrated with higher values than 10A****
const float m2= 10.0/(I_DC300_RawAt10A - I_DC300_RawAt0A); //gradient of mapping adc output (x) to current (y)
const float c2= -m2*I_DC300_RawAt0A; // y-intercept for graph of current (y) vs adc output (x)

void setup() {
  // Start serial communication for debugging purposes
  Serial.begin(115200);

  //start MCP3008 connection
  mcp.begin(_SCLK, _MISO, _MOSI, _SS);

  //Connect to Wifi
  WiFiSetup();

  //set Up SD card (check for card, write headings, prepare for writing values etc.)
  setUpSDCard();

  //Set up RTC with laptop time if first time using.
  setUpRTC();

  ThingSpeak.begin(client); // Initialize ThingSpeak

  //Set up ADC sensors
  setUpACensors();
}

void loop() {
  // Increment readingID on every new reading
  readingID++;
  getReadings();
  getTimestamp();
  logSDCard();
  updateThingSpeak();
  delay(5000);
}

// Function to update all ADC readings
void getReadings(){
  updateACValues();
  updateDCValues();
}

void WiFiSetup(){
  // Connect to Wi-Fi network with SSID and password
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected.");
}

void setUpRTC(){
  if (! rtc.begin()) {
    Serial.println("Couldn't find RTC");
    Serial.flush();
    abort();
  }

  if (! rtc.lostPower()) {
    Serial.println("RTC is NOT running, 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));
  }
}

// Function to get date and time from RTC module
void getTimestamp() {
  //get time and date from RTC module
  rtc.now().toString(dateStamp);
  rtc.now().toString(timeStamp);
}

//set Up SD Card. Write data headings if new file.
void setUpSDCard(){
  // Initialize SD card
  SD.begin(SD_CS);
  if(!SD.begin(SD_CS)) {
    Serial.println("Card Mount Failed");
    return;
  }
  uint8_t cardType = SD.cardType();
  if(cardType == CARD_NONE) {
    Serial.println("No SD card attached");
    return;
  }
  Serial.println("Initializing SD card...");
  if (!SD.begin(SD_CS)) {
    Serial.println("ERROR - SD card initialization failed!");
  }
}

```



```

    return;    // init failed
}

// If the data.txt file doesn't exist
// Create a file on the SD card and write the data labels
File file = SD.open("/data.txt");
if(!file) {
    Serial.println("File doesn't exist");
    Serial.println("Creating file...");
    writeFile(SD, "/data.txt", "ReadingID, dayStamp, timeStamp, I_DC50, I_DC300, V_DC50, V_DC1000, I1_AC100, I2_AC100, I3_AC100, V_AC500 \r\n");
} else {
    Serial.println("File already exists");
}
file.close();

// Write the sensor readings on the SD card
void logSDCard() {
    SD.begin(SD_CS);
    if(!SD.begin(SD_CS)) {
        Serial.println("Card Mount Failed");
        return;
    }
    uint8_t cardType = SD.cardType();
    if(cardType == CARD_NONE) {
        Serial.println("NO SD card attached");
        return;
    }
    Serial.println("Initializing SD card...");
    if (!SD.begin(SD_CS)) {
        Serial.println("ERROR - SD card initialization failed!");
        return;    // init failed
    }
    dataMessage = String(readingID) + "," + String(dateStamp) + "," + String(timeStamp) + "," +
        String(I_DC50) + "," + String(I_DC300) + "," + String(V_DC50) + "," + String(V_DC1000) + "," + String(I1_AC100) +
        "," + String(I2_AC100) + "," + String(I3_AC100) + "," + String(V_AC500) + "\r\n";
    Serial.print("Saving data to SD Card: ");
    Serial.println(dataMessage);
    appendFile(SD, "/data.txt", dataMessage.c_str());
}

// Write to the SD card (DON'T MODIFY THIS FUNCTION)
void writeFile(fs::FS &fs, const char * path, const char * message) {
    Serial.printf("Writing file: %s\n", path);

    File file = fs.open(path, FILE_WRITE);
    if(!file) {
        Serial.println("Failed to open file for writing");
        return;
    }
    if(file.print(message)) {
        Serial.println("File written");
    } else {
        Serial.println("Write failed");
    }
    file.close();
}

// Append data to the SD card (DON'T MODIFY THIS FUNCTION)
void appendFile(fs::FS &fs, const char * path, const char * message) {
    Serial.printf("Appending to file: %s\n", path);

    File file = fs.open(path, FILE_APPEND);
    if(!file) {
        Serial.println("Failed to open file for appending");
        return;
    }
    if(file.print(message)) {
        Serial.println("Message appended");
    } else {
        Serial.println("Append failed");
    }
    file.close();
}

void updateThingsSpeak(){
    //set what we want each ThingsSpeak fields and statuses to be updated to
    ThingsSpeak.setField(1,I_DC50);
    ThingsSpeak.setField(2,I_DC300);
    ThingsSpeak.setField(3,V_DC50);
    ThingsSpeak.setField(4,V_DC1000);
    ThingsSpeak.setField(5,I1_AC100);
    ThingsSpeak.setField(6,I2_AC100);
    ThingsSpeak.setField(7,I3_AC100);
    ThingsSpeak.setField(8,V_AC500);
    ThingsSpeak.setStatus("no dump load triggered");
    // write to the ThingsSpeak channel
    int x = ThingsSpeak.writeFields(myChannelNumber, myWriteAPIKey);
    if(x == 200){
        Serial.println("ThingsSpeak Channel update successful.");
    }
    else if(x == -401){
        Serial.println("Failed to write to ThingsSpeak. Most probable cause is the rate limit of once every 15 seconds exceeded");
    }
    else{
        Serial.println("Problem updating channel. HTTP error code " + String(x));
    }
}

// Callback method for reading the pin value from the MCP instance (needed for EmonLib)
int MCP3008PinReader(int _pin){
    return mcp.analogRead(_pin);
}

void setUpACsensors(){
    emon1.inputPinReader = MCP3008PinReader; // Replace the default pin reader with the customized ADC pin reader
    emon2.inputPinReader = MCP3008PinReader;
    emon3.inputPinReader = MCP3008PinReader;

    emon1.voltage(V_AC500_Pin, V_AC500_Calibration, V_AC500_Phase); // Voltage: input pin, calibration, phase_shift
    emon1.current(I1_AC100_Pin, I1_AC100_Calibration); // Current: input pin, calibration.
    emon2.voltage(V_AC500_Pin, V_AC500_Calibration, V_AC500_Phase);
    emon2.current(I2_AC100_Pin, I2_AC100_Calibration);
    emon3.voltage(V_AC500_Pin, V_AC500_Calibration, V_AC500_Phase);
    emon3.current(I3_AC100_Pin, I3_AC100_Calibration);
}

void updateACValues(){
    emon1.calcVI(20,2000); // Calculate all. No.of half wavelengths (crossings), time-out
    emon2.calcVI(20,2000);
}

```

```

emon3.calcVI(20,2000);

RealACPower1      = emon1.realPower;      //extract Real Power into variable
ApparentACPower1   = emon1.apparentPower;   //extract Apparent Power into variable
ACPowerFactor1     = emon1.powerFactor;     //extract Power Factor into Variable
RealACPower2      = emon2.realPower;
ApparentACPower2   = emon2.apparentPower;
ACPowerFactor2     = emon2.powerFactor;
RealACPower3      = emon3.realPower;
ApparentACPower3   = emon3.apparentPower;
ACPowerFactor3     = emon3.powerFactor;

V_AC500           = emon1.Vrms;             //extract Vrms into Variable
I1_AC100          = emon1.Irms;             //extract Irms into Variable
I2_AC100          = emon2.Irms;
I3_AC100          = emon3.Irms;
}

void updateDCValues() {
  //read ADC, convert to voltage, and scale by calibration factor
  V_DC50 = (mcp.analogRead(V_DC50_Pin))*(float)Vref*(float)V_DC50_Calibration/(float)adcLevels;
  V_DC1000 = (mcp.analogRead(V_DC1000_Pin))*(float)Vref*(float)V_DC1000_Calibration/(float)adcLevels;

  //read ADC, and scale using y= mx+c with calibration constants 'm' and 'c'
  I_DC50 = ((mcp.analogRead(I_DC50_Pin))*m1) +c1;
  I_DC300 = ((mcp.analogRead(I_DC300_Pin))*m2) +c2;
}

```

14.4 Interesting Similar Projects

By studying existing energy monitoring projects, a lot can be learnt on better/alternative ways of collecting and analysing data. The following projects are a selection of those that were found during the research phase:

14.4.1 Expandable 6 Channel ESP32 Energy Meter Main Board [15]

This open-source project uses the same ESP32 backbone as our sensing projects, and 6 AC current sensors, identical to those in our sensor. There is a well-developed user interface for reading and recording measurements to the cloud. The code is available on GitHub and if time allows, could be integrated to improve the user experience of our monitoring device.

14.4.2 Home Energy Monitor (v2) [11]

ESP32-based home energy monitor consisting of a single AC current sensor, using code from the ‘OpenEnergyMonitor’. They use the Amazon Web Server to send information to the cloud, DynamoDB to store the data online, and GraphQL to visualize the data.

14.4.3 PV Module Monitoring System Based on Low-Cost Solutions [16]

A wireless low-cost PV-module monitoring system based on open-source solutions, using two Arduino UNO’s and a Raspberry-Pi.

14.4.4 A Project to Design and Build a Controller to Route Surplus P.V. Energy into the Domestic Hot Water Supply [13]

Has information on how to trigger a dump load using a Triac. The system is from open OpenEnergyMonitor and designed for solar installations that are connected to the grid, aiming to operate a dump load when there is surplus solar power being produced that is being fed into the grid. This is slightly different to our application with charge controllers and solar batteries.

15 References

- [1] MIDI-CHIP, “MCP3008,” GitHub, [Online]. Available: <https://github.com/MIDI-CHIP/MCP3008>. [Accessed 01 03 2021].
- [2] ESP-Marius, “SD Card,” GitHub , [Online]. Available: https://github.com/espressif/esp-idf/tree/master/examples/storage/sd_card#note-about-gpio12. [Accessed 02 03 2021].

- [3] C. Stratton, "Why put a resistor in series with signal line?," Stack Exchange - Electrical Engineering, 10 12 2010. [Online]. Available: <https://electronics.stackexchange.com/questions/7709/why-put-a-resistor-in-series-with-signal-line#:~:text=Adding%20a%20resistor%20to%20a,to%20the%20chip's%20supply%20current..> [Accessed 15 03 2021].
- [4] Robojax, "Measuring 50A to 200A using Allegro ACS770 Current Sensor with Robojax Arduino Library," YouTube, 05 04 2020. [Online]. Available: <https://www.youtube.com/watch?v=sB6EULTix2k> . [Accessed 15 03 2021].
- [5] Glynhudson, "EmonLib," OpenEnergyMonitor GitHub, [Online]. Available: <https://github.com/openenergymonitor/EmonLib>. [Accessed 18 01 2021].
- [6] KlausST, "AC Voltage Measurement using PIC," EdaBoard, 08 06 2016. [Online]. Available: <https://www.edaboard.com/threads/moved-ac-voltage-measurement-using-pic.355515/#post-1520931>. [Accessed 09 03 2021].
- [7] "HCNR200 and HCNR201 High-Linearity Analog Optocouplers, datasheet," AVAGO technologies.
- [8] R. Marston, "Optocoupler circuits," Nuts & Volts Magazine, 02 2000. [Online]. Available: <https://www.nutsvolts.com/magazine/article/optocoupler-circuits>. [Accessed 16 03 2021].
- [9] "CT Sensors - Interfacing with an Arduino," Open Energy Monitor, [Online]. Available: <https://learn.openenergymonitor.org/electricity-monitoring/ct-sensors/interface-with-arduino>. [Accessed 10 01 2021].
- [10] P. Wieland, "EmonLib," GitHub, 16 02 2018. [Online]. Available: <https://github.com/PaulWieland/EmonLib/tree/4a965d87061c19ad8b0a35bb173caada014ecbd9>. [Accessed 30 03 2021].
- [11] "DIY Home Energy Monitor: ESP32 + CT Sensors + Emonlib," Savjee Simply Explained, [Online]. Available: <https://savjee.be/2019/07/Home-Energy-Monitor-ESP32-CT-Sensor-Emonlib/>. [Accessed 08 03 2021].
- [12] "ESP32 Data Logging Temperature to MicroSD Card," Random Nerd Tutorials, [Online]. Available: <https://randomnerdtutorials.com/esp32-data-logging-temperature-to-microsd-card/>. [Accessed 02 03 2021].
- [13] "Choosing an Energy Diverter," Learn OpenEnergyMonitor, [Online]. Available: <https://learn.openenergymonitor.org/pv-diversion/introduction/choosing-an-energy-diverter.md>. [Accessed 08 03 2021].
- [14] "High Voltage Arc Gap Calculator," [Online]. Available: <https://www.cirris.com/learning-center/calculators/50-high-voltage-arc-gap-calculator?>. [Accessed 10 03 2021].
- [15] "Expandable 6 Channel ESP32 Energy Meter Main Board," CircuitSetup, [Online]. Available: <https://circuitsetup.us/index.php/product/expandable-6-channel-esp32-energy-meter/>. [Accessed 08 03 2021].
- [16] A. M.-A. G. S.-N. M. B. a. Á. M.-G. J. Paredes-Parra, "PV Module Monitoring System Based on

-] Low-Cost Solutions: Wireless Raspberry Application and Assessment,” *Energies*, vol. 11, no. 11, p. 3051, 2018.
- [17 B. Schewber, “How should grounds and commons be connected to each other?,” EE World Online, 20 o3 2017. [Online]. Available: <https://www.eeworldonline.com/grounds-commons-connected/>. [Accessed 15 03 2021].