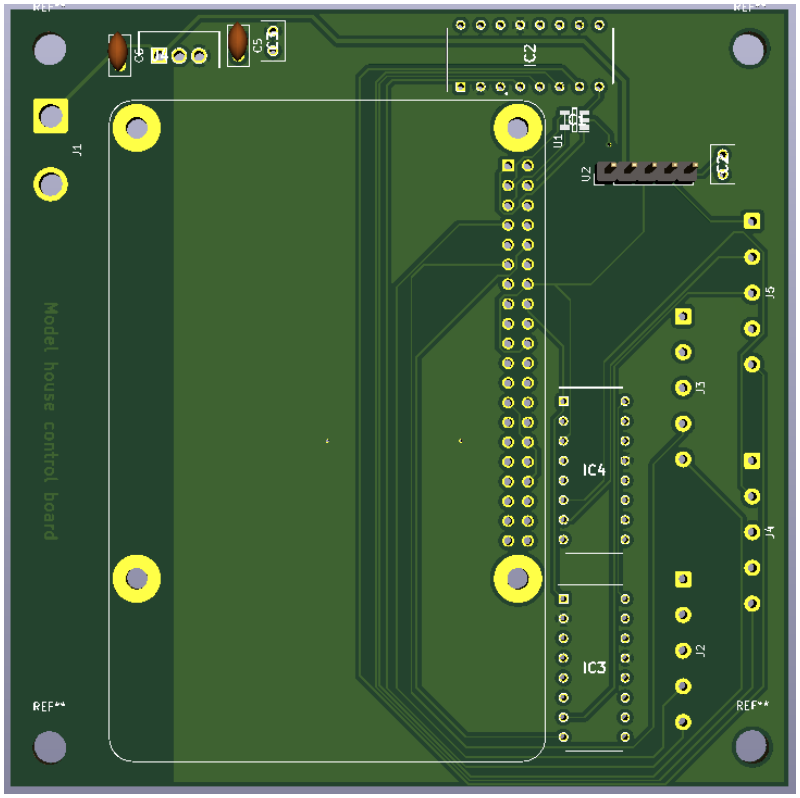
FAN Summer Scholarship Report:

DC Microgrid Outreach Project Control System



Andre Webber-LaHatte

Introduction:

The aim of this project is to produce a scale model DC microgrid containing 4 houses. The main function of this project is to develop a product that will display the power consumption of a house over a 24-hour period. This will work by assigning a list of appliances to each house. The appliances will be represented by an LED placed somewhere inside a scale model house, and a load resistor, chosen to dissipate power at an approximately 1:100 scale. A program will read a CSV file containing the on and off times for each appliance over a 24-hour period, and run a simulation. The total simulation time should be between 1 and 5 minutes long, with each tick representing 1 minute of time. During the simulation, the current to the load resistors will be sampled so that a power vs time graph can be produced during the simulation. The supply to the house will be 24V with a maximum power output of 100W. Through this I have gained useful insights and developed skills in electronics and design. This report will describe the design process and operating principle of the DC microgrid control system.

Design Approach:

My design involved building off a system that could control 1 house, to design a system that could control 4 houses. This design used a Raspberry Pi as a base module and a Pi Pico as a sub module to control houses. A DIN cable connection is used between the main control board with the raspberry Pi, and the house board with the Pico. Upon understanding the work that had already been done, I broke the system down into 3 sub-systems:

Communication:

> This system will enable the Raspberry Pi to communicate individually with each Pico.
>
> The single house communication system used UART to send a list of on and off states of each appliance, called a state vector. Each state vector would be sent only when that transition is to be made. Testing of this system showed that for simulation times of 2 minutes or less, consecutive transition messages would be received as a single UART message causing switch states to be skipped.
>
> The approach I took to this subsystem was one of time management. The communication time should be sufficiently short so that the Pi Pico can distinguish between consecutive switch messages for simulation times as low as one minute. The switch messages should also allow for sufficient time after switching for sampling the new current.

Switching:

> This system will be used to switch the appliances on and off.
>
> The single house switching system made use of 3 CD4094BE 8-bit shift registers in a parallel configuration. The Pi Pico would load the on/off values onto these registers. According to the CD4094BE datasheet, the outputs can source up to 3.6mA with a 5V supply, this is not enough current for each LED, so the shift register outputs were used to switch TN2106N3-G MOSFETs which were used to switch both LEDs and load resistors.
>
> I felt this system needed some improvement. I focussed on the time aspect of this, with consideration given into allow the Pico to do as little as possible between a switching message, and outputting the next state vector onto the shift registers.

Current Detection:

This system will detect and record the current for graphing power over time.

The single house current detection system used the ACS712 hall effect current sensor. The output of the sensor fed into an MCP2641 OP-AMP positive gain circuit used to 'spread' the voltage output of the sensor so that the expected current range would result in voltages from 0V-5V. The OP-AMP then output to MCP3002 10-bit ADC. The ADC was located on a different board to the load resistors

Neither the OP-AMP nor the current sensor were available to me when I took over this project. My approach to this system was to ensure the previous system could be implemented mostly the same based on availability of selected components. I also considered whether ADC had a sufficient bit depth to detect the smallest expected changes in current.

Designs and Operation:

Communication:

The first step in improving the communication system was to shrink the data size. The state vectors being sent in the original design were a list of integers for each appliance, each being a 1 or 0 to indicate on or off. The same amount of information can be sent using a single integer for each 8 appliances. State vectors were generated by reading a CSV file where each row indicated an appliance, on or off, and a list of times where this 'command' would occur. Once read, the program would store the information as a dictionary that associates a list of commands with each time. By associating hexadecimal values, and on/off indicators with each command, state vectors can be generated by performing logical ANDs and ORs on the previous state vector, using 0x00 to generate the first. Off commands would use ANDs operations with hex numbers where all bits are HIGH except the one being flipped, and vice versa for on commands. State vectors are split into separate integers (one for each 8 appliances) using logical ANDs with a 0xff mask, and bit shifting. The entire list of state vectors is generated prior to the start of the simulation. Figure 1 shows a block diagram of this program.

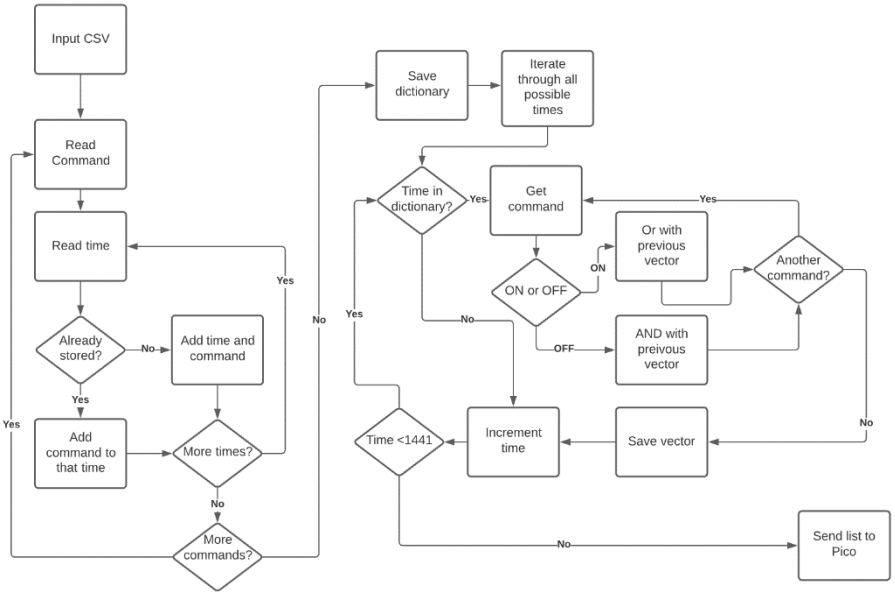


Figure 1: Process for generating an ordered list of state vectors from reading an input CSV

Testing with a smaller data size showed a simulation for a single could now be done in one minute, however as 4 houses need to be run, this is still too much time. Rather than send state vectors one at a time, this system was changed to send all state vectors corresponding to a house prior to the simulation. The control board indicates a required switch transition to the Pico by pulling a line low. Two 4:1 multiplexers will be used so the Pi can address houses for receiving and transmitting over UART. My design implements an 8:1 multiplexer as I had thought the requirement was for 8 houses and not 4. As another system requires the use of an analogue multiplexer, the ADG608BNZ analogue multiplexer was used in this case to keep component selection consistent. Figure 2 shows a schematic for this part of the circuit.

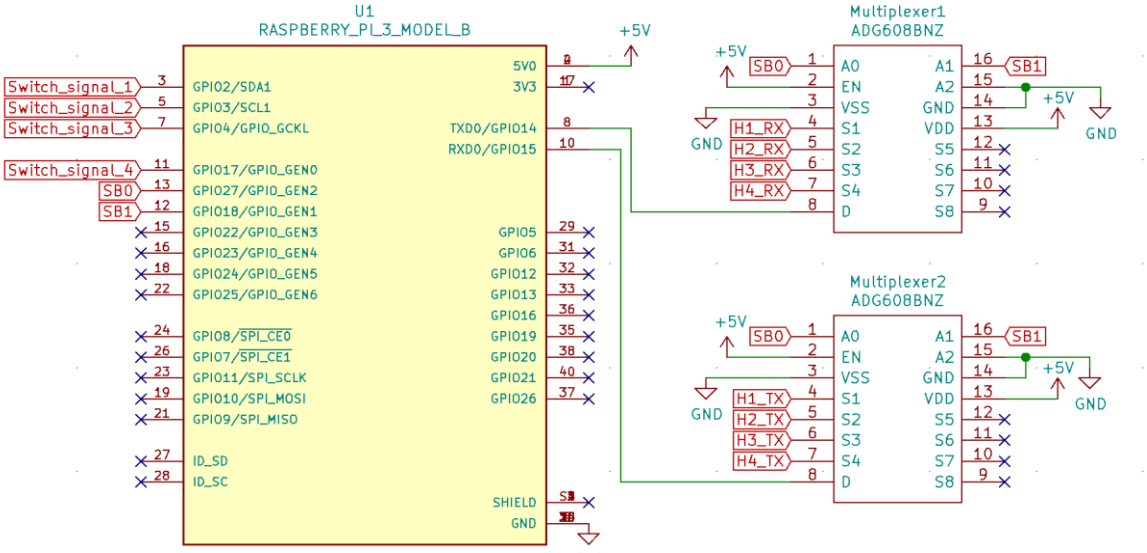


Figure 2: Schematic showing Raspberry Pi using multiplexers for “addressable” UART, and individual “switch signal” lines for each house.

Switching:

As stated above, switching was achieved using an 8-bit shift register. The CD4094BE operates by clocking serial data onto an internal register, then uses a ‘strobe’ to transfer the data in parallel to the output. The original designs for a single house had 3 registers all using a separate data, clock, and strobe lines for each register, however the datasheet shows a serial configuration can allow for 3 registers to essentially be used as a single 24-bit register., saving the use of 6 GPIO pins on the Pi Pico. Figure 3 shows this configuration.

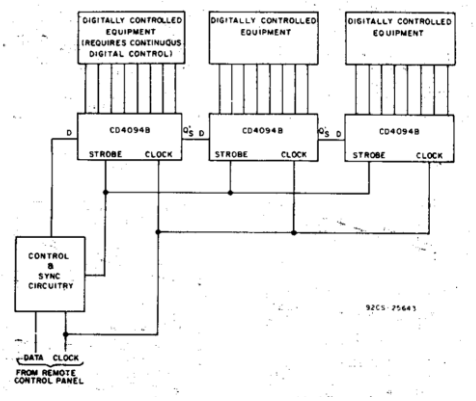


Figure 3: CD4094BE in series configuration

The previous design had each value loaded onto the internal register and strobed to the output at each transition. A more efficient approach allows switch transition to occur much faster. The internal registers allow state vectors to be preloaded before being output. Upon receiving a signal to switch, the Pico then immediately strobes the stored value onto the output, then preloads the next value. This greatly minimises the switching time, freeing up more time for current sampling.

The software to do this starts with processing the list of state vectors received from the Pi. The Pico uses a combination of bit shifting and OR operations to combine each integer group into a single word. Values are loaded onto the register by bit shifting each vector by a decrementing amount, and loading this value ANDed with a mask of value 1 onto the data output to the register. The bit shift needed to decrement as the registers would load values as MSB first. When the Pi signals to the Pico a transition needs to be made the strobe line will be pulled LOW, then the next value loaded onto the internal register. Figure 4 shows a block diagram of this process.

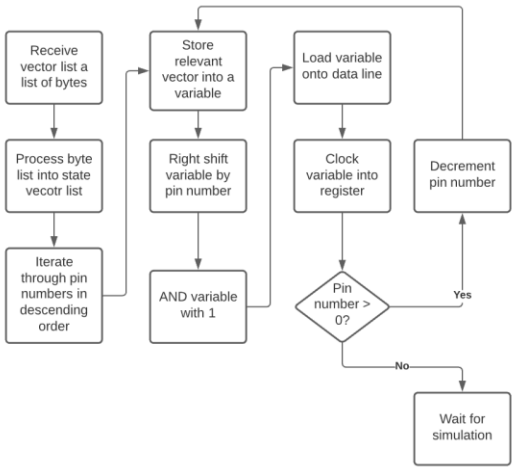


Figure 4: Process for sorting byte list to vector list, and loading first vector onto internal register

Lastly was the switching device. For the LEDs, rather than using individual MOSFETs for each, TLC59211 8-bit DMOS Sink Drivers can be used. The datasheet indicates this IC can sink up to 200mA. The LEDs I have selected have an optimal operating current of 20mA, so 200mA is sufficient. This device will output a low when the corresponding input is high. Figure 5 shows a diagram of how these sink drivers should be configured with the CD4094BE, resistors, and LEDs. For the load resistors I chose to use the IRLB8721PBF MOSFET, the datasheet stating a typical $R_{DS(on)}$ of $6.5m\Omega$ and a maximum V_{DS} of 30V. This on resistance will have a negligible impact on the current to each load resistor, and the rated voltage is higher than what is being applied.

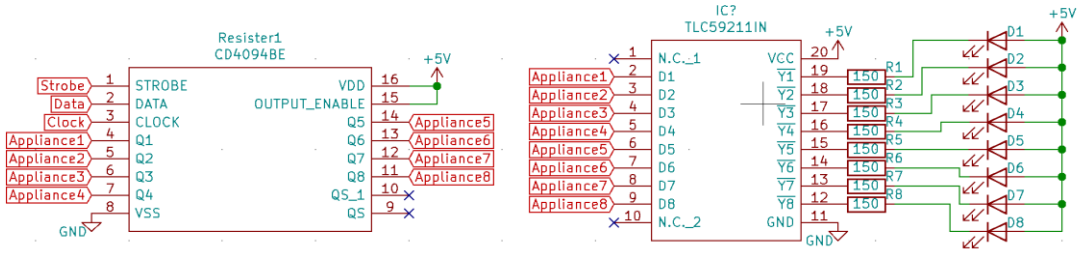


Figure 5: Schematic showing configuration of registers with sink drivers to power LEDs

Current Detection:

Current detection faced me with the most challenges. The current sensor chip used in the original design was not available, which required the selection of a different chip and the design of a new OP-AMP circuit. Ideally, the design would have implemented an omni directional current sensor however no supplier had these in stock. The datasheet for the ACS723 current sensor states a 400mV/A output ratio. Having one of these chips available, I decided to test the functionality of this chip before designing an OP-AMP circuit. As advised by a technician, this testing was done by shorting a bench power supply through the ACS723 chip and used the current limiter to control the current. I took several measurements in increments of 50mA up to 1A, 100mA up to 2A, and 200mA up to 3A. The data showed a linear relationship between voltage and current, with a gradient of 400mV/A. This gave me confidence that any OP-AMP circuit I design would bring the output of the ACS723 to a desired range. ***Figure 6 shows a graph of the current sensor test data.***

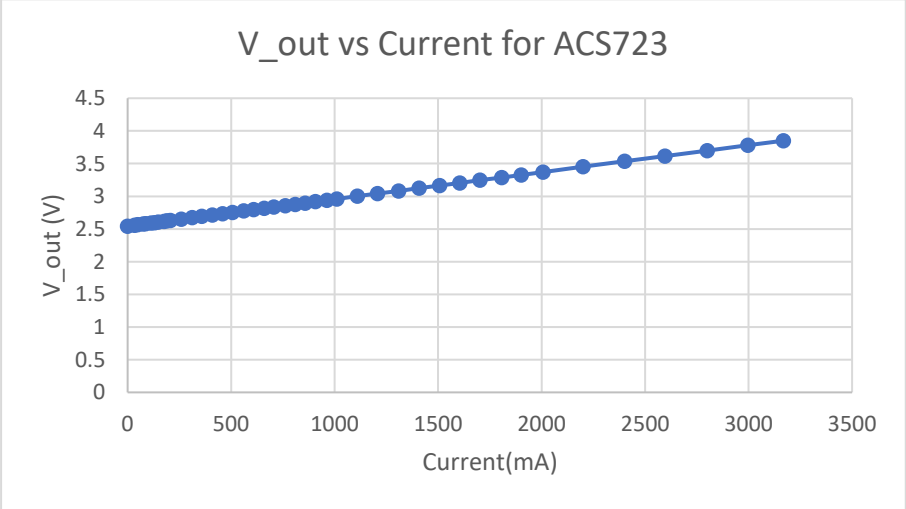


Figure 6: Graph of current sensor test data

The bi-direction functionality of this chip resulted in an output of 2.5V at 0A with a 5V supply. Ideally the current sensor should output 0V at 0A to best work with sampling the output using a single ended ADC. The solution to this was to supply the ACS723 with $\pm 2.5V$. Using a voltage divider to output 2.5V to an MCP2641 OP-AMP in buffer configuration**.** The buffer amplifier output to the positive supply pin on the ACS723, as well as the input to the LM2662 charge pump switching regulator in a voltage inversion configuration to output -2.5V, ***see figure 7.*** The datasheet for the LM2662 states testing was done with $47\mu F$ capacitors with ESR bellow $200m\Omega$. I selected the Nichicon UCX1V470MCL1GS for this.

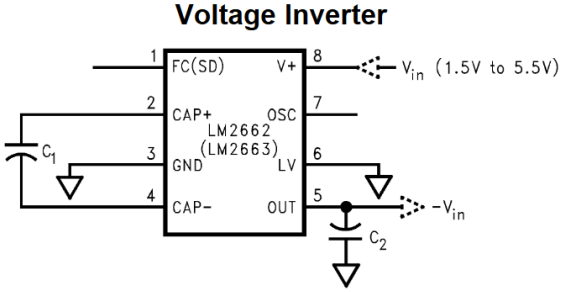


Figure 7: LM2662 in the voltage inversion configuration

For sampling the output voltage of the ACS723, I selected the MCP3221 12-bit sample and hold ADC. Based on 400mV/A, and the calculated current range, the output voltage range of the ACS723 should be 0V-1.67V, an amplification factor of 2 is required to "spread" this range to a 0V-3.3V that will work with the MCP3221 ADC. 3.3V was used rather than 5V as the MCP3221 communicates via I^2C, for which the raspberry pi used 3.3V as HIGH.

The MCP3221 ADC has a conversion rate of 22.3k samples per second which is far higher than I need. A single MCP3221 can sample all 4 houses well within a single tick of a 1-minute simulation. The voltage output from each current sensor amplifier circuit is output to the ADG608BNZ analogue multiplexer, which then outputs to the MCP3221. The MCP3221 datasheet states an analogue input states the source impedance should be kept as low as possible, ideally within the 10s of ohms to adhere to the $1.12\mu s$ analogue input acquisition time. The core resistance measured in the DIN cables I had on hand was 1Ω, and the ADG608BNZ states an on resistance of 40Ω. The source impedance to the ADC is approximately 41Ω which is low enough to give me confidence the distance from the current sensor amplifier output to the ADC will have a negligible impact on the accuracy in data acquisition. ***Figure 8 shows this circuit configuration.***

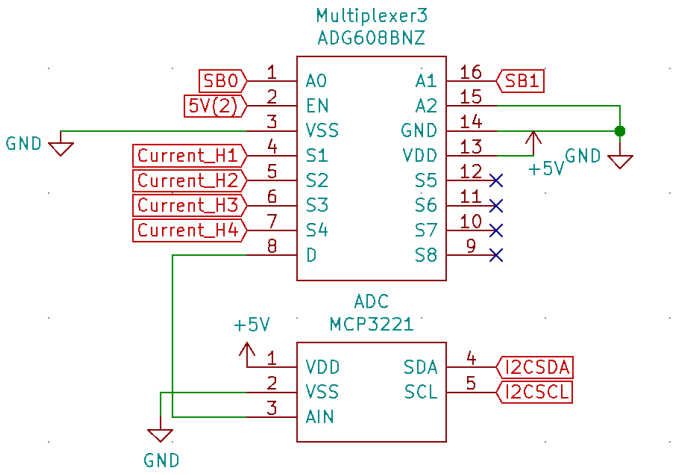


Figure 8: Using an analogue multiplexer to read 4 different voltages with one ADC

Conclusion:

The design process I used enabled me to produce a control system for a DC microgrid. I have produced designs for five PCBs, one for running the simulation, and the other four to control the switching in each house. I have programmed the control to run the simulation, while live graphing the data it is sampling.

I am disappointed I was not able to get as far with this project as I had hoped, however I gained some good insights. The information I have taken from completing this project has deepened my understanding of the principles of electronics and design. The largest challenge I faced was learning how to trust myself to make the correct decision. This was most apparent when designing the current detection circuit which required considering many approaches, then researching and selecting components to achieve the desired result. Other aspects such as communication required reasonable understanding of each protocol to be set up correctly. Through this I have learned to approach projects through learning and planning, rather than solely trial and error, as I have tended

to do in the past. Overall, this project has been an exercise in continued learning, and has inspired me to change the direction of my studies to focus more on electronics and control systems.

Appendix:

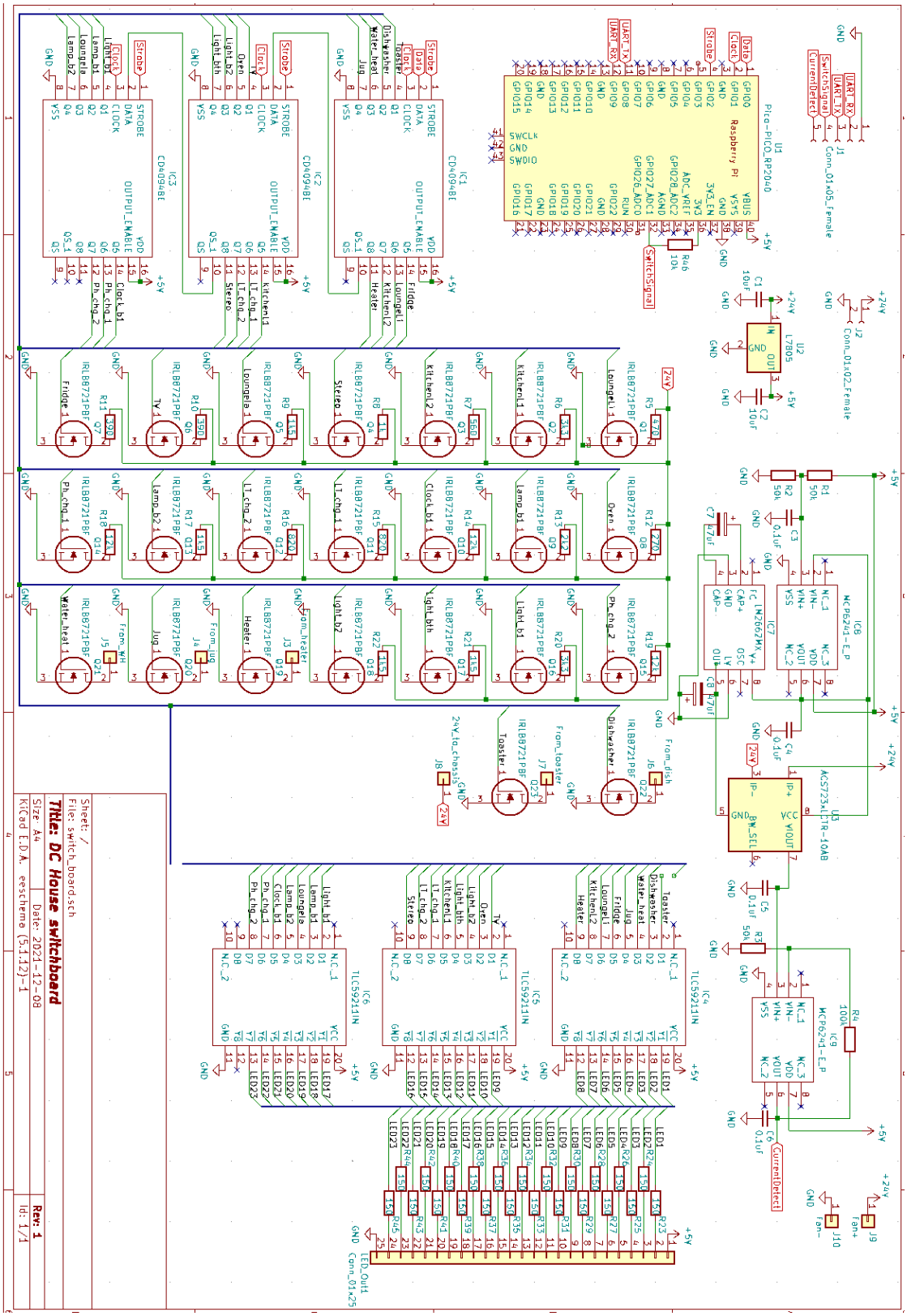


Figure 1: Full schematic for a single switch board

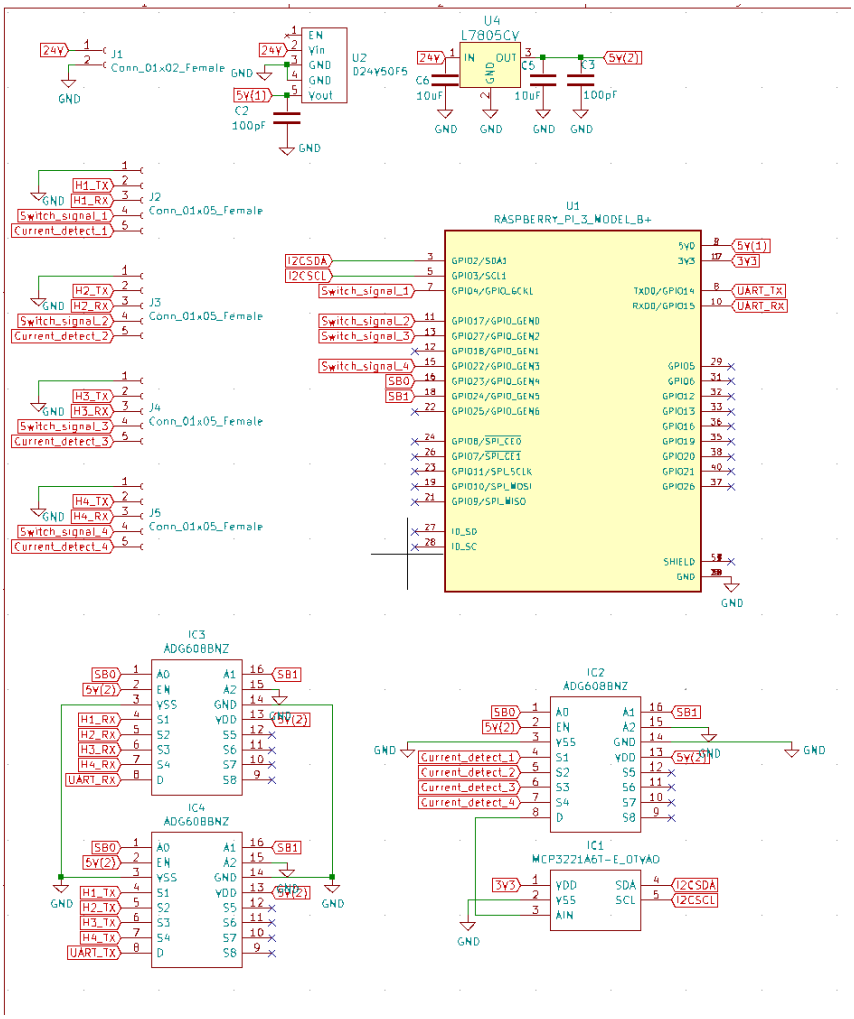


Figure 2: Full schematic of control board

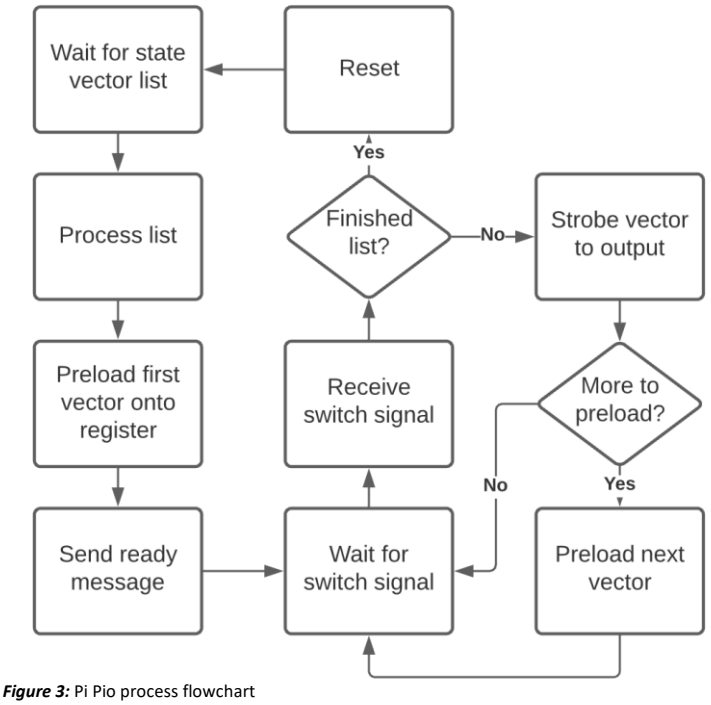


Figure 3: Pi Pio process flowchart

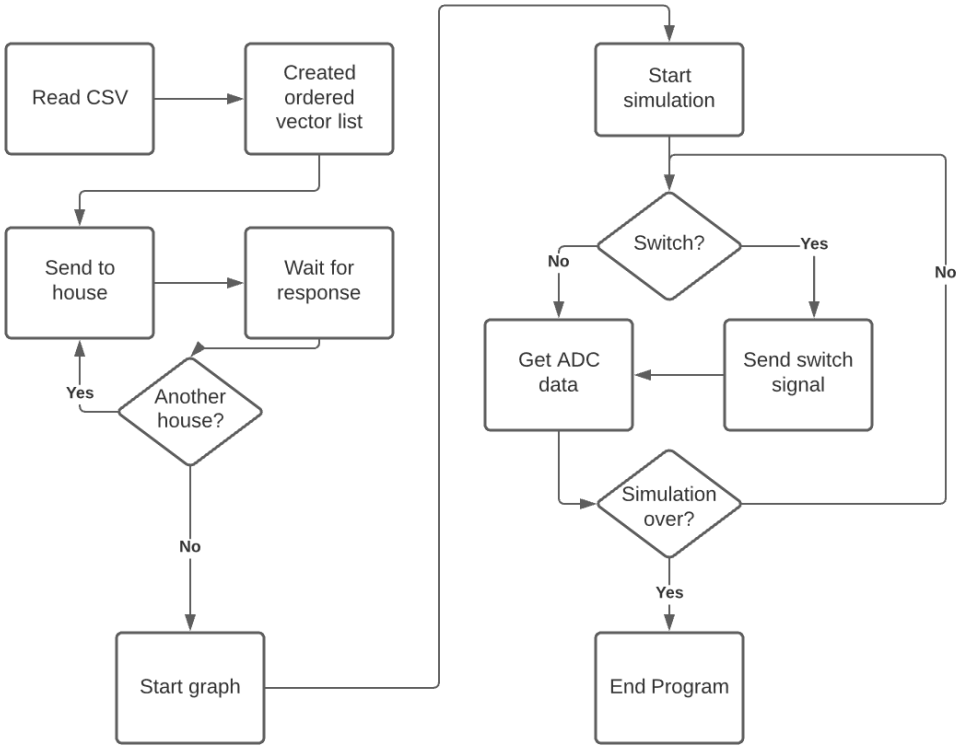


Figure 4: Raspberry Pi process flow chart