Phase B Documentation

Alex Kondracki
6/10/2023
ENCE 3220 Embedded Systems 2023 DU

Project Requirements

The goal of this project is to build a kitchen timer. The timer is powered by an ATmega32U4-A controller, the same as the Arduino Uno R3. The timer can be communicated with via an ESP. The timer needs to have a start/stop and increment button. It needs to display the time on a 7-segment display and it needs to make a noise when the time reaches 0. All of these components must fit onto a custom designed PCB no bigger than 100 x 100 mm. Also the 328P is powered by USB and programmed by ISP.

The exact requirements are as follows:

- Uses ATmega328P
- Fits on a 100 x 100 mm PCB
- Displays time accurately on a 7-segment display
- Has a start/stop button.
- Has an increment button.
- All UI elements are accessible via wifi over the ESP
    - Can read/write data from the ESP
- Can be programed via ISP
- Powered via USB

1. System Design

The system is set up to be controlled by an ATmega32U4-A. The 32U4-A can be written to and is powered using micro usb. An ESP is used for UART communication. Two buttons with status leds are set up to toggle and increment the timer. A buzzer is included as well. For the display a 7-segment display and shift register are used in tandem to display the time.

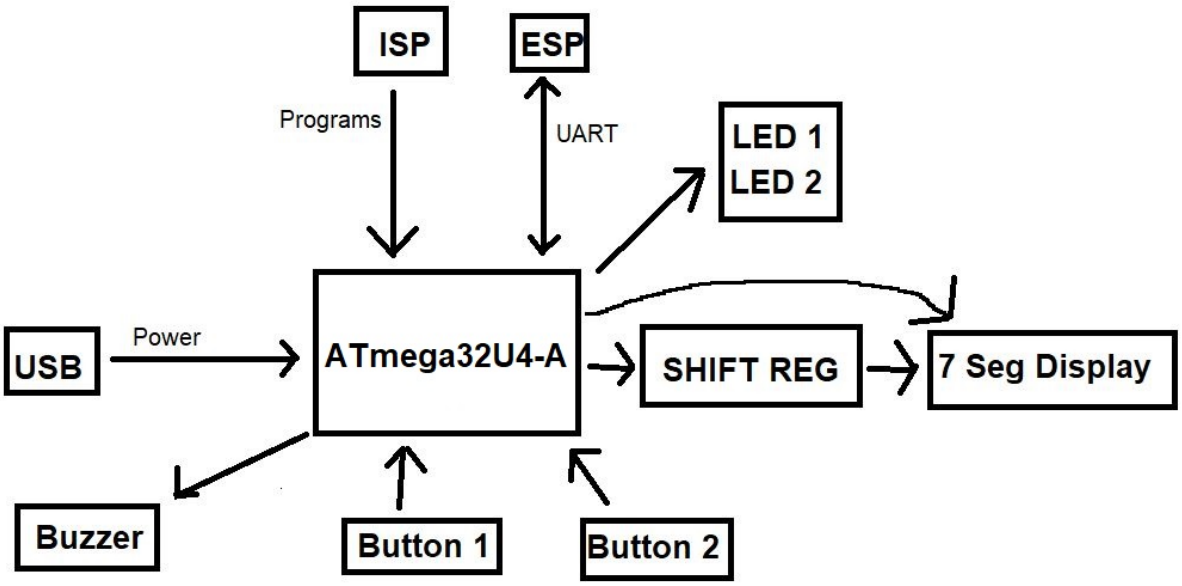

Figure 1. Basic Block Diagram of Timer

2. Components Selection

The components selected are as follows:

Number	Value	Footprint	Quantity
1	0.1uF	C_0603_1608Metric	8
2	22p	C_0603_1608Metric	2
3	1uF	C_0603_1608Metric	2
4	10uF	C_0805_2012Metric	1
5	2.2uF	C_0603_1608Metric	1
6	10nF	C_0603_1608Metric	1
7	100	R_0805_2012Metric	8
8	10k	R_0805_2012Metric	2
9	330	R_0805_2012Metric	2
10	10k	R_0603_1608Metric	2
11	22	R_0603_1608Metric	2
12	1k	R_0603_1608Metric	1
13	LED	LED_0805_2012Metric	3
14	CA56-12EWA	CA56-12EWA	1
15	74HC595	TSSOP-16_4.4x5mm_P0.65mm	1
16	ATmega32U4-A	TQFP-44_10x10mm_P0.8mm	1
17	USBLC6-2SC6	SOT-23-6	1
18	LP2985-3.3	SOT-23-5	1
19	16MHz	Crystal_SMD_Abracon_ABM8G-4Pin_3.2x2.5mm	1
20	PTCSMD	Fuse_1812_4532Metric	1
21	PTS125SM43SMTR21M_LFS	PTS125_SMD_Button	2
22	Speaker	Buzzer_12x9.5RM7.6	1
23	PTS526_SM08_SMTR2_LFS	PTS526_SMD_Button	1
24	AVR-ISP-6	PinSocket_2x03_P2.54mm_Vertical	1
25	USB_B_Mini	USB_Mini-B_Lumberg_2486_01_Horizontal	1
26	ESP_Conn	PinSocket_2x04_P2.54mm_Vertical	1

Table 1. Components by Footprint. These can be viewed in PCB_Design\Phase_B_ATMEGA_v3\BOM\ibom.html

3. Build Prototype

The prototype is built around an ATMEGA. It has a 16 Mhz clock and reset switch. A mini USB provides the board with power and is connected to the controller via ESD protection. Additionally a row of capacitors exist in parallel to debounce the power supply. Using a voltage regulator a ESP8266 is wired to the controller. The buzzer is Piezoelectric which means it has no moving parts (a current stimulates a sheet to bend.) The 7 segment display is enabled/disabled by the controller, with a shift register to write values. There are two buttons to control the timer and status leds to match each button.

4. PCB Design

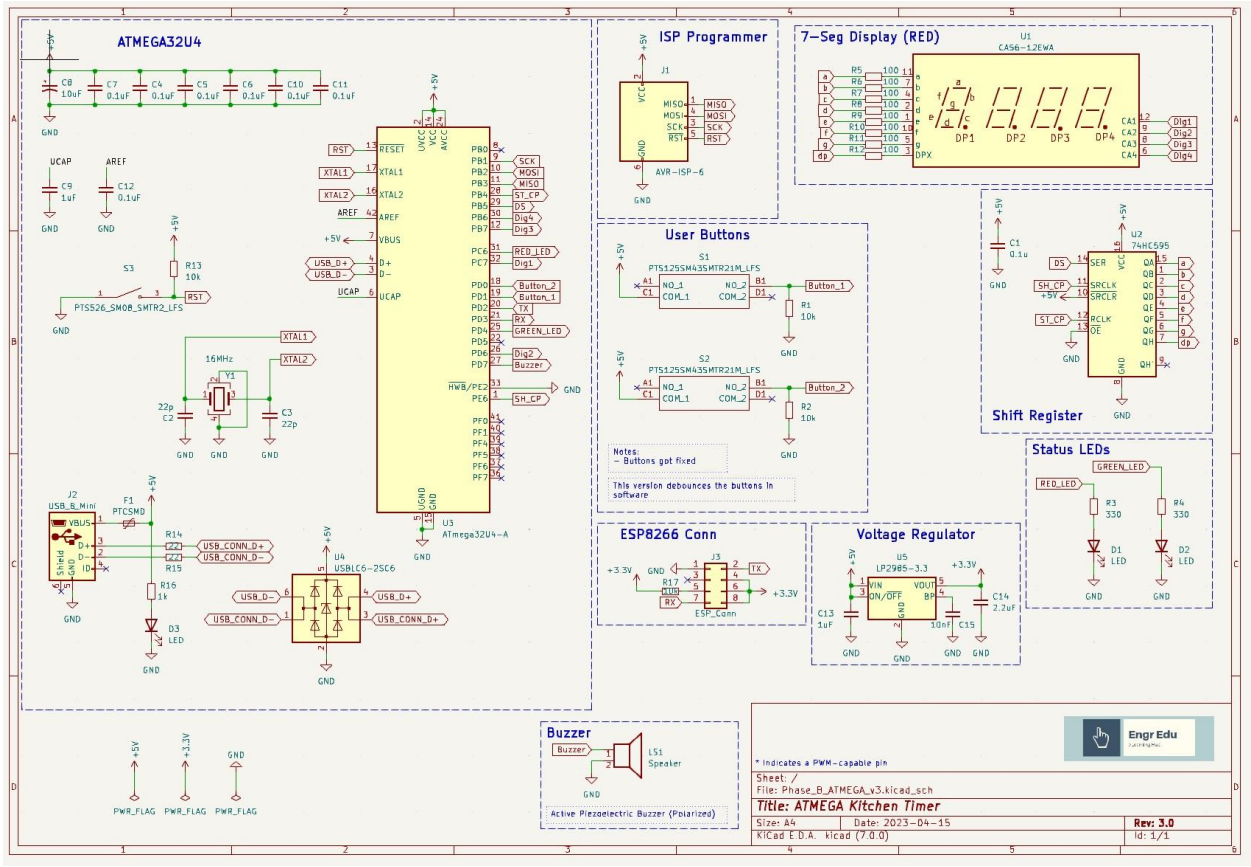

Figure 2. Wiring Schematic. Refer to sections 2,3 for part descriptions.

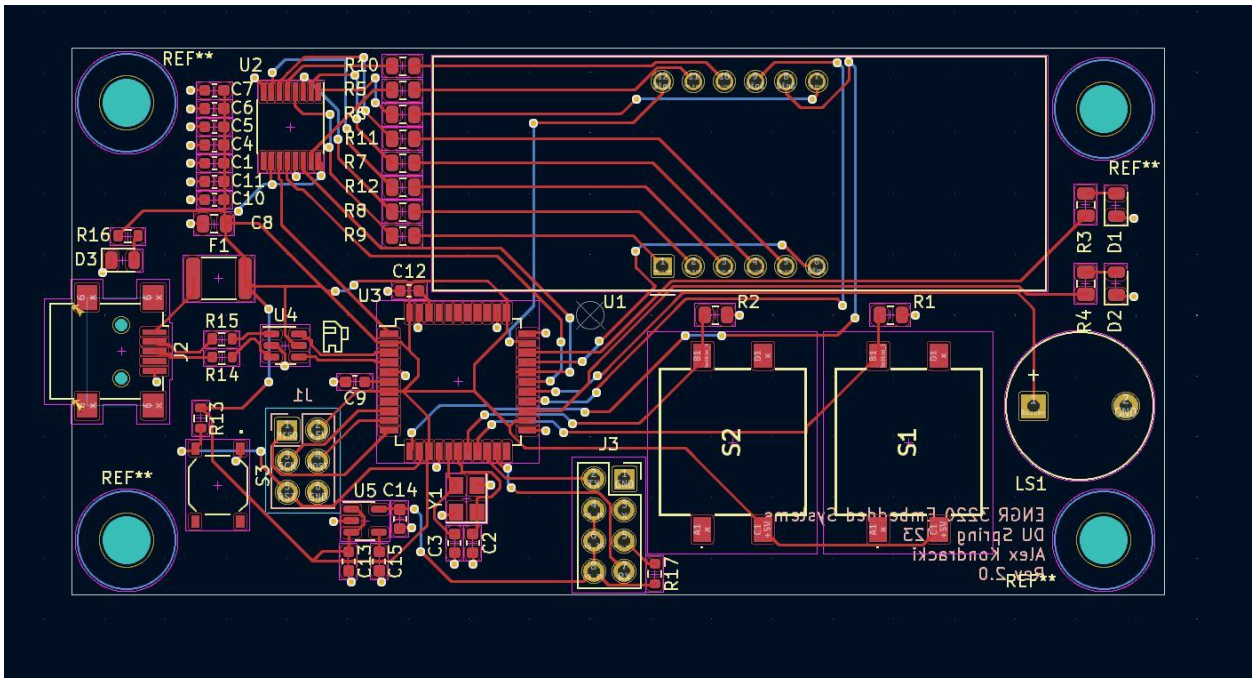

Figure 3. PCB Schematic. Description Below.

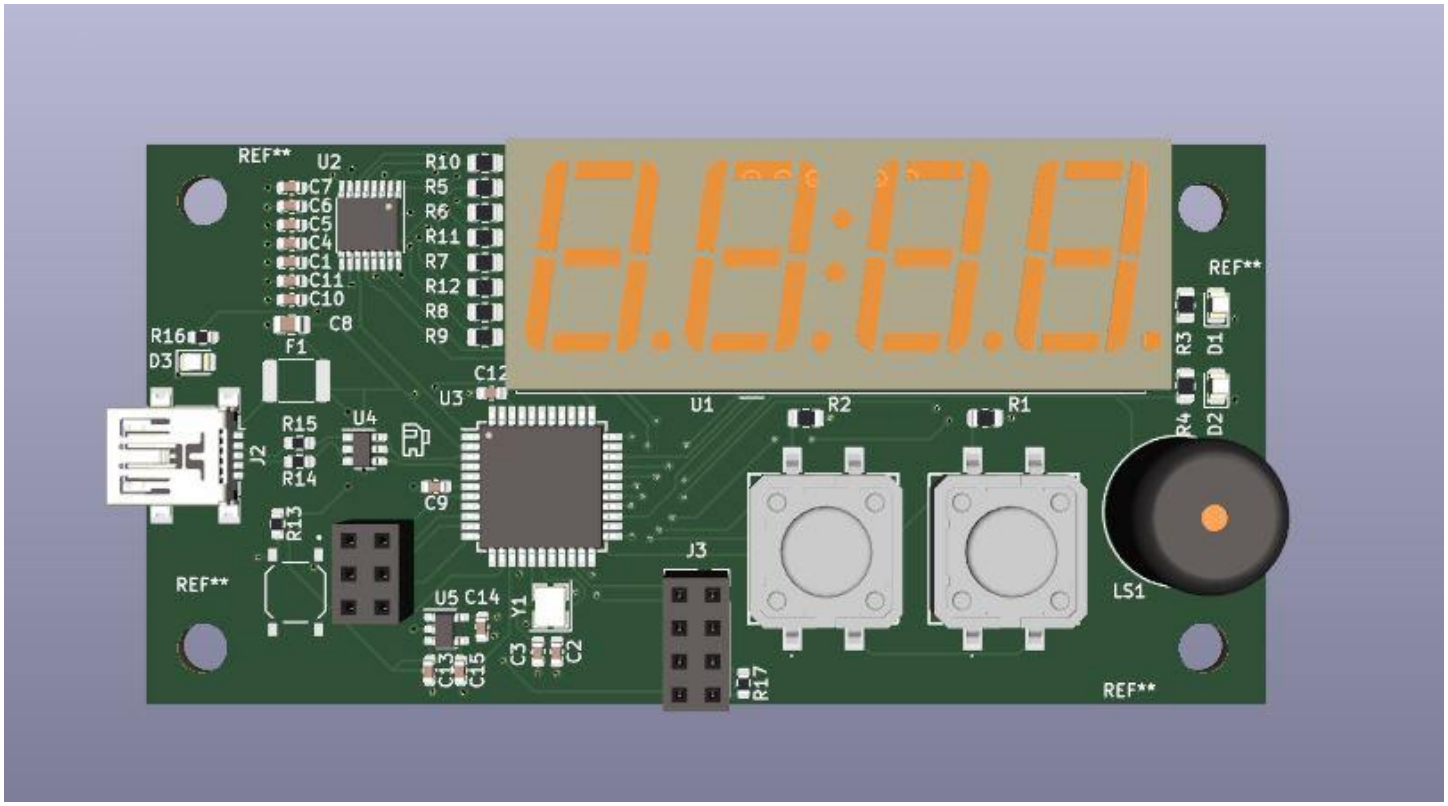

Figure 4. PCB Render. Description below

The PCB is set up with the ATMEGA in the center. The mini USB on the right provides power to the board. It goes through an ESD connector(U4) to connect to the controller. Above U4 is a series of capacitors to debounce power. Bottom Left is the reset button and slightly to the right is the ISP programmer. As the isp programmer has inverted footprint the footprint appears flipped and the J1 label is on the adjacent side. Directly below the ATMEGA is Y1 which is the 16 MHz clock. On the left and right of the clock is the Voltage Regulator and ESP8266 respectively. Near the top left is the shift register U2 which rights values to the 7 segment display U1. The status leds are in the far right and buttons/buzzer are in the bottom right.

Note that there are two flaws with this layout. The holes are equidistant from the sides (see Notes\pcb_dimensions.jpg). The capacitors should be closer to the ATMEGA to prevent stability issues. Finally the ESP is not in a good location. With the shape of the ESP8226 (J3,) the component now has to mount under the board to not be in the way of the button. This required modification of the case.

5. Assemble Stage

Soldering the components to the board took roughly 6 hours. After the assembly was “complete” there were a few issues that had to be sorted out. The first was that Button 1, 2 and RX on the microcontroller were shorted. This was not initially caught as the short was not visible from above the circuit. Upon observing the ATMEGA pins from the side a hidden glob of solder was revealed to be connecting all three pins. The second was the top line of the 7 segment display was not working. This was fixed by adding more solder to R6; the resistor responsible for that part. Third, the top of the reset button fell off. This was due to cleaning the board with isopropyl alcohol and was fixed by re-adding kapton tape over the top. The only other problem is the front two solder pads on the Mini USB port have detached from the pcb making the port less stable. This has not been fixed.

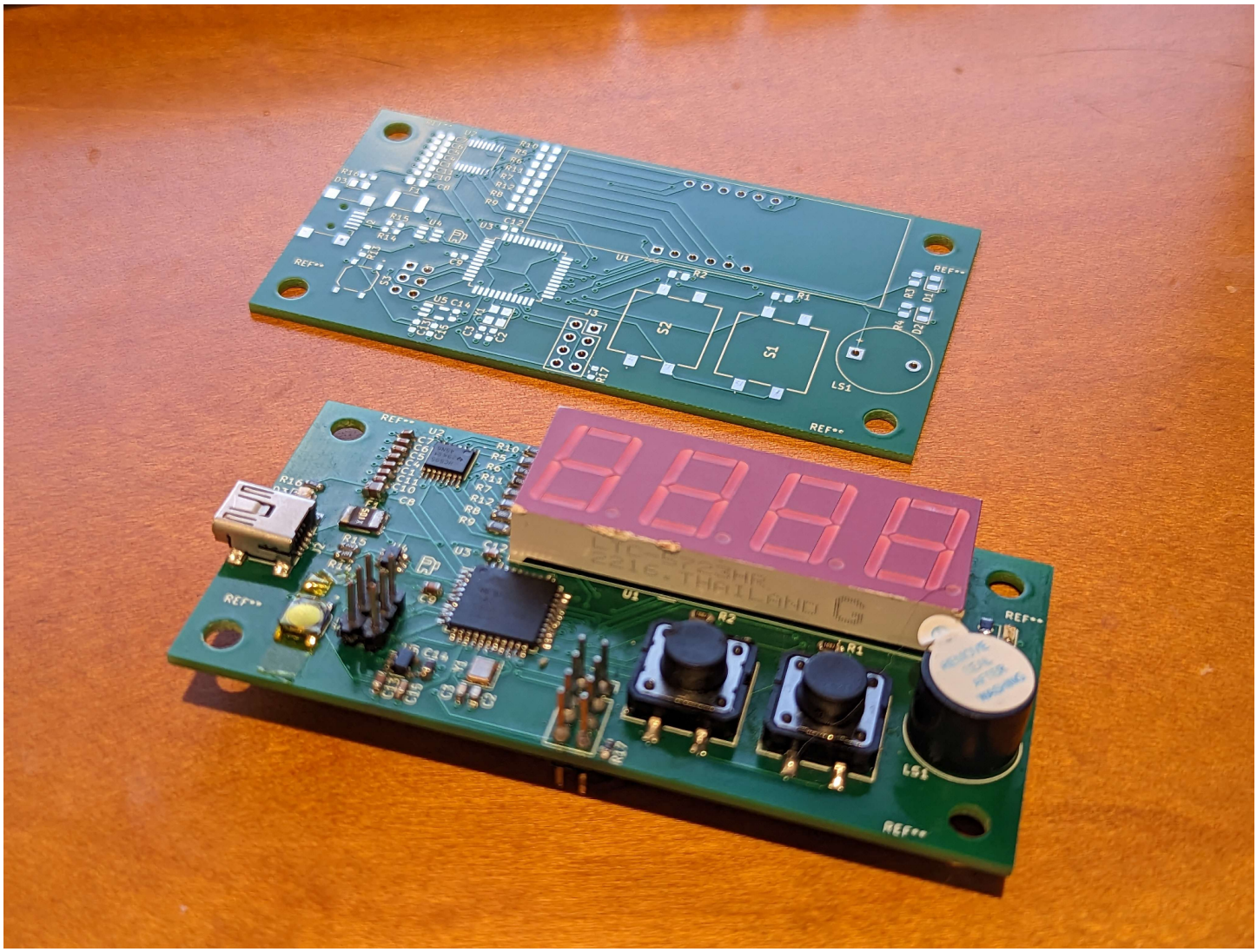

Figure 5. Before and after shots. The ESP (normally found connected to the right set of pins,) was removed due to difficulties in testing.

6. Software Development

The code is set up using a modular style. It uses separate header and C files (Disguised as Cpp to please the IDE,) for each section of code. The main file defines overall timings and all the commands. The I/O file provides functions for interaction with all I/O. The COM file provides incoming data parsing. Lastly the time file provides the layout for a timer and all functions to interact with it. This is so it is simple to control the timer with either incoming commands or I/O.

The io file contains a setup and io interacting functions. The only notable interaction is the shift register and 7 segment display. The correct pin of the display is enabled shift register shifts the desired value to display, after the display pin is disabled. After rotating through each digit, the time is displayed.

The timer is driven by a tick function which represents a single tick of the timer. This function only runs when the timer is enabled. When a counter reaches a certain number of ticks it is considered one second and the timer decreases. If the timer reaches 0 the timer is disabled and the end state is enabled. The increment function increments the time, disables the end state regardless of the timer state. The toggle function swaps the timer state as long as the time is above 0, then it disables the end state.

The parse file calls a function every tick if there are enough incoming characters to read. It then checks if the command starts with a '$' char. After it compares the incoming message char by char against all possible values, if a value doesn't match it is not tested for the remainder of the function. If a command is read, a flag is raised and the command variable is set. Otherwise the command is set to NUL.

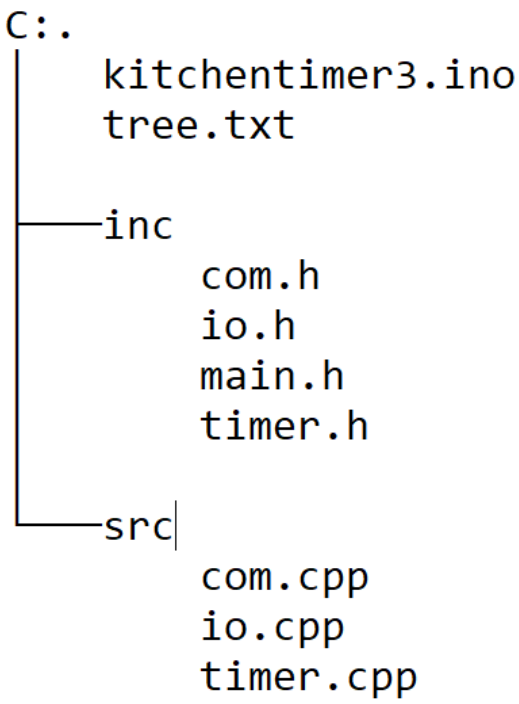

Figure 6. File layout for code

```
Block Diagram

Main
    Setup
        Clock to 16MHz
        IO Setup
        ISR Setup
        IO Init
        Serial Init

Loop
    Check Timer Flag
        tick timer
        tick com (Serial Parsing)
    Check Button 1
        toggle timer
    Check Button 2
        inc timer
    Check Command
        Case
            NULL: nothing
            INC: inc timer
            STP: if on toggle timer
            STR: if off toggle timer
        clear command
    Display Time
    Write the end state to Buzzer
    Write the end state to RED LED
    Write the timer state to GREEN LED
```

Figure 7. Code diagram. Note that all statuses (excluding command) are flags raised by interrupts.

7. Enclosure Design

The enclosure is very simple. It is an open shell for the PCB. Four cylinders provided a mounting point for the PCB to rest on the case. The hole is due to the ESP8266 needing to be connected from the bottom to avoid interference for the buttons. The enclosure took 3 hours to print and worked after sanding even with a cylinder missing. The ESP had enough clearance above the ground.

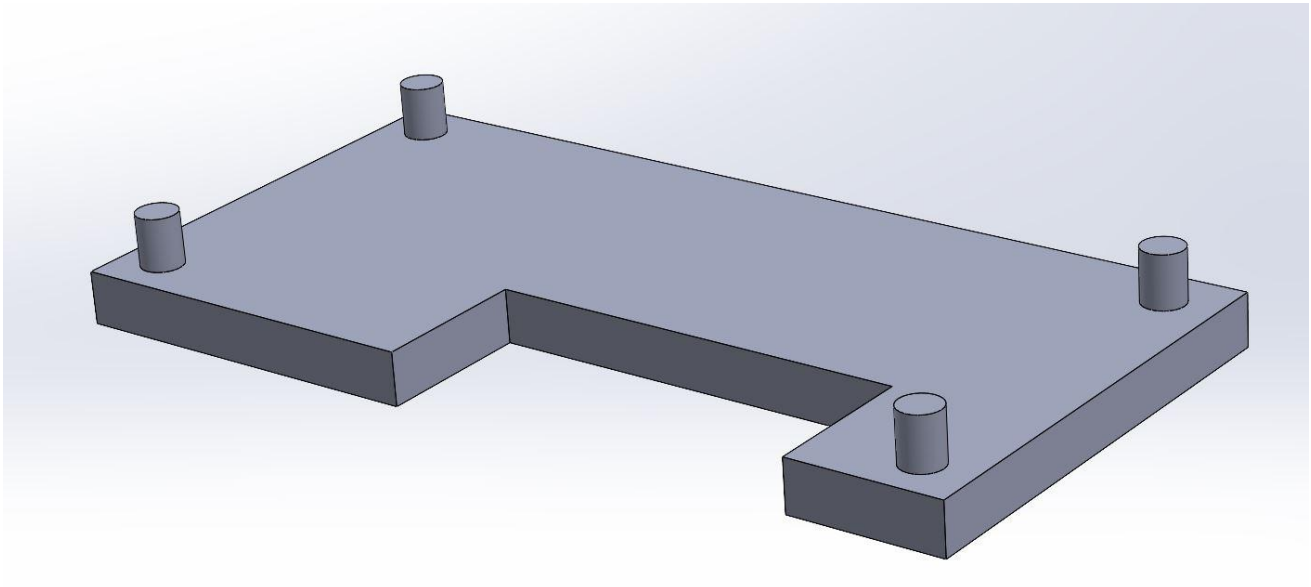

Figure 8. Render of the enclosure.

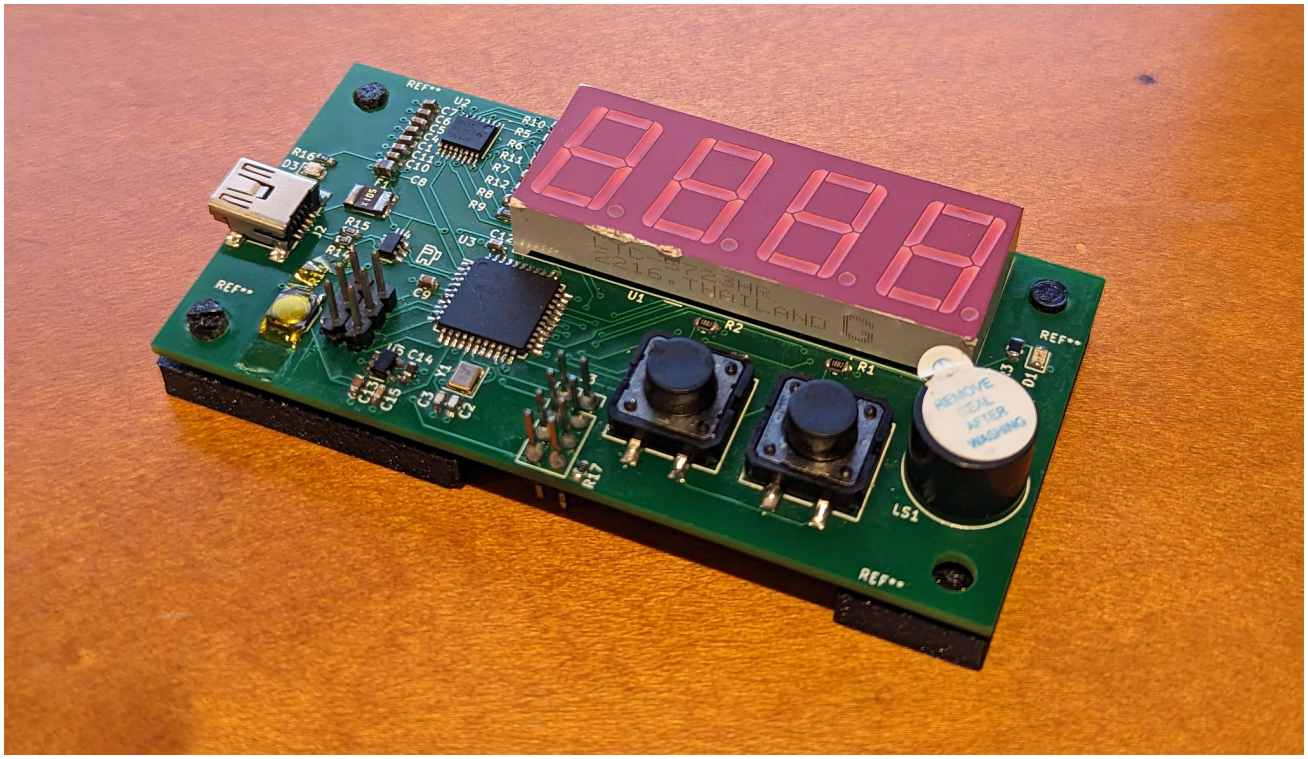

Figure 9. Picture of the enclosure and PCB.

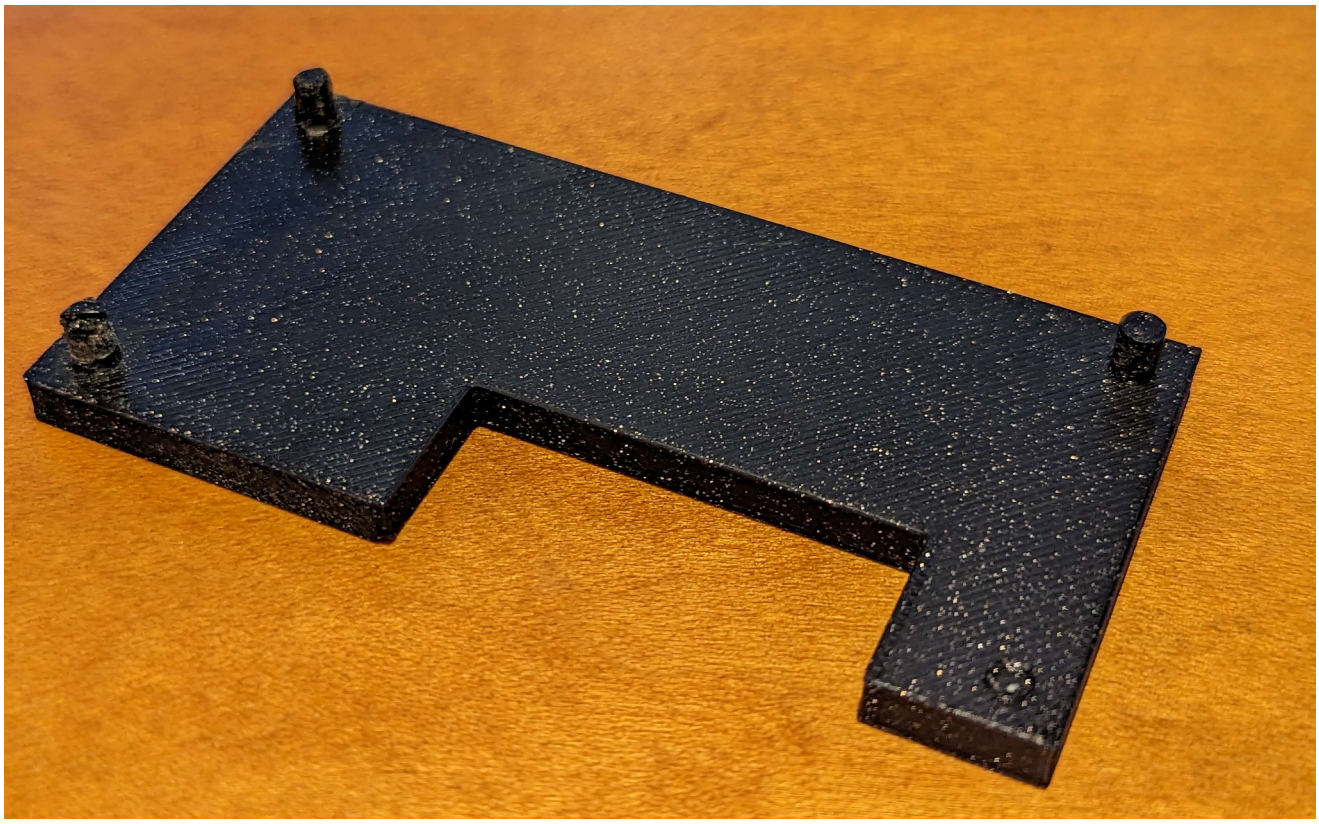

Figure 10. Picture of the enclosure.