Phase B Documentation

Alex Kondracki
5/4/2023
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.

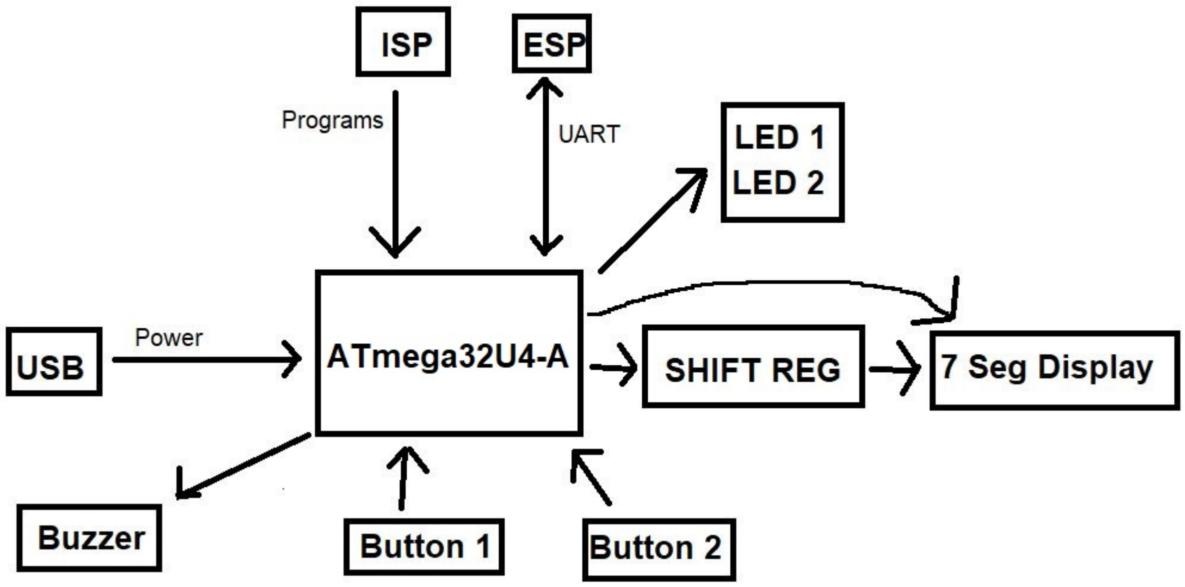


Figure 1. Basic Block Diagram of Timer

2. Components Selection

The components selected are as follows:

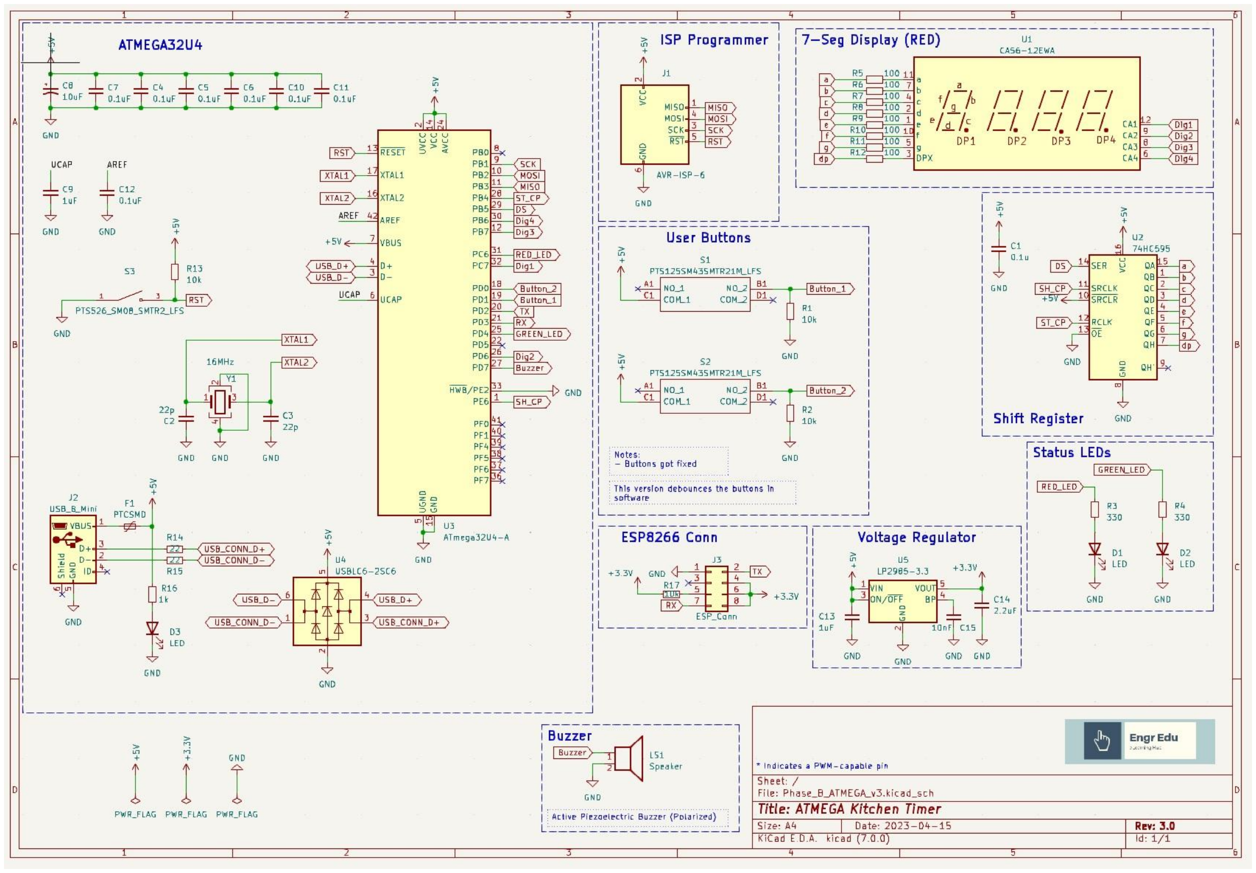
Number	Value	Footprint	Quantity
1	0.1uF	C_0603_1608Metric	8
2	22p	C_0603_1608Metric	2
3	1uF	C_0603_1608Metric	2
4	10uF	C_0805_2012Metric	1
5	2.2uF	C_0603_1608Metric	1
6	10nF	C_0603_1608Metric	1
7	100	R_0805_2012Metric	8
8	10k	R_0805_2012Metric	2
9	330	R_0805_2012Metric	2
10	10k	R_0603_1608Metric	2
11	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



ATMEGA32U4
ISP Programmer
7-Seg Display (RED)
User Buttons
Shift Register
Status LEDs
ESP8266 Conn
Voltage Regulator
Buzzer
Notes: - Buttons got fixed
This version debounces the buttons in software
Active Piezoelectric Buzzer (Polarized)
* Indicates a PWM-capable pin
Sheet: /
File: Phase_B_ATMEGA_v3.kicad_sch
Title: ATMEGA Kitchen Timer
Size: A4
Date: 2023-04-15
KiCad E.D.A. kicad (7.0.0)
Rev: 3.0
Id: 1/1
Engr Edu

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

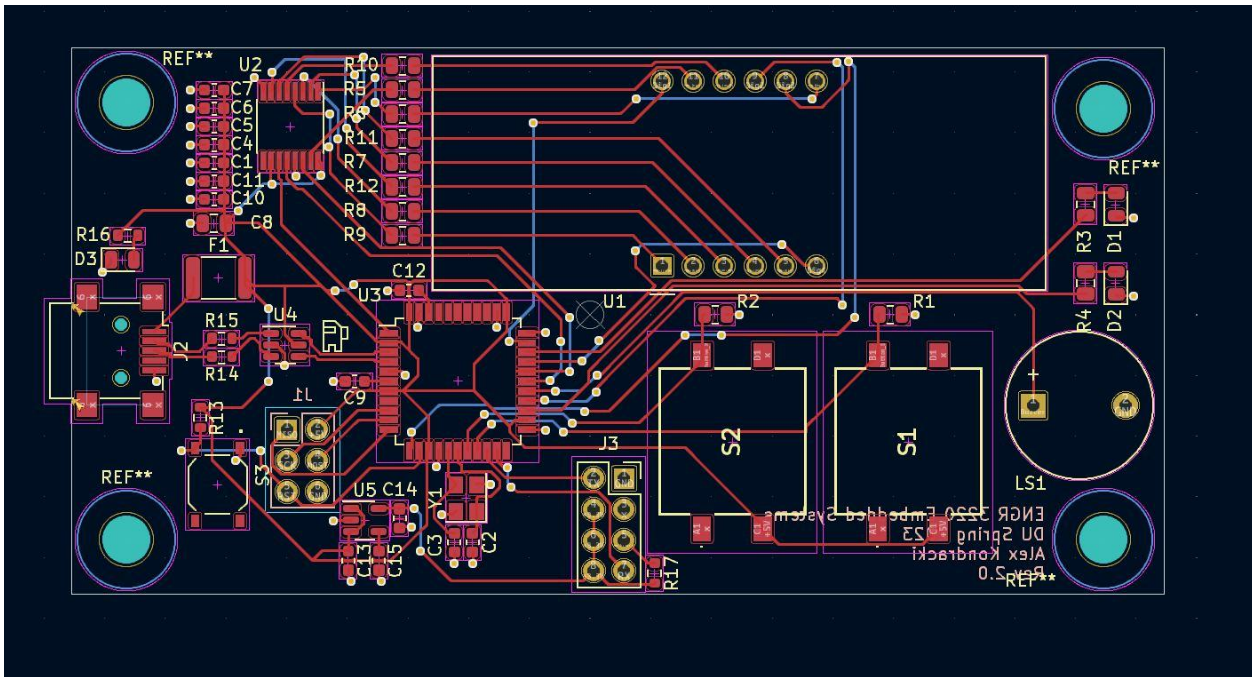


Figure 3. PCB Schematic. Description Below.

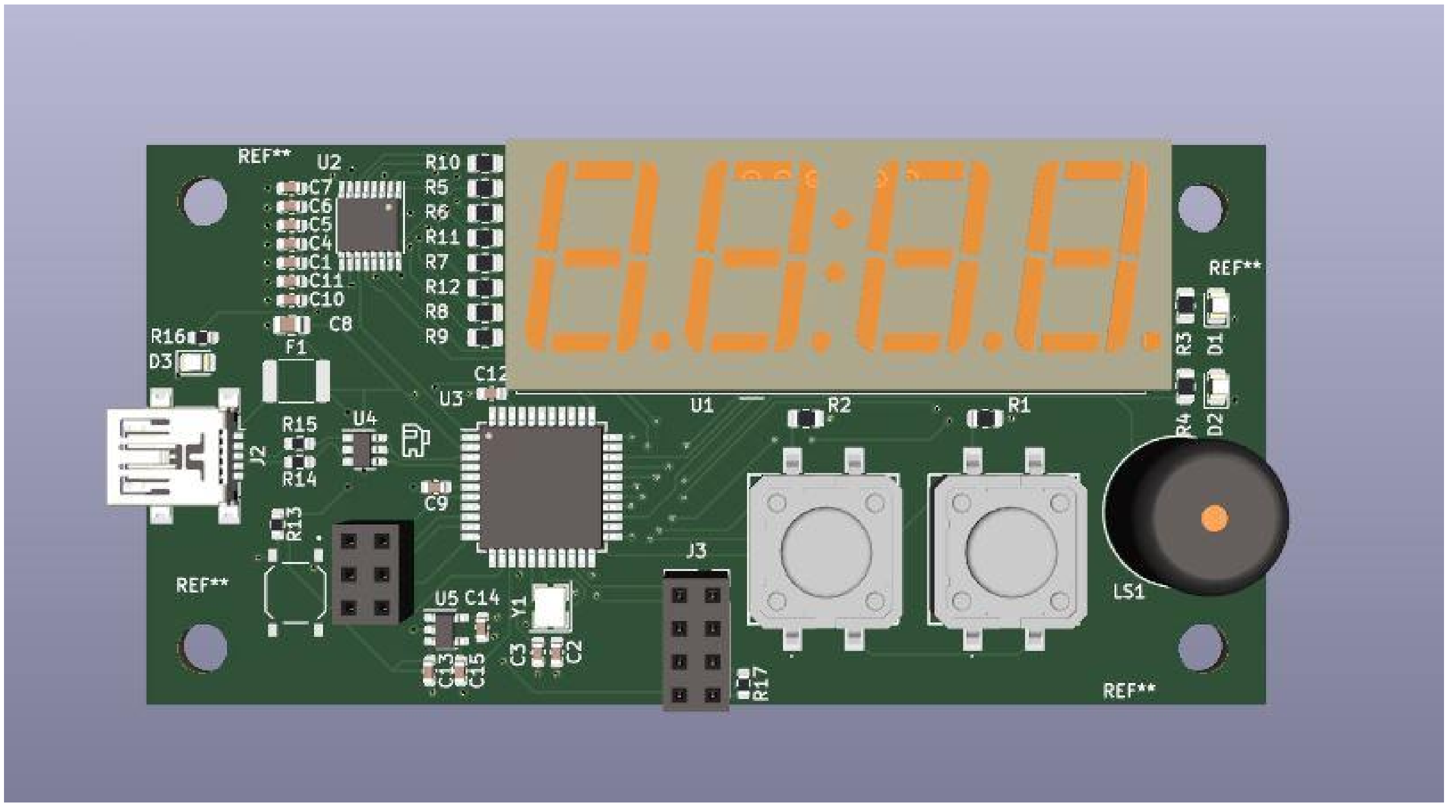


Figure 4. PCB Render. Description below

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 a 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.

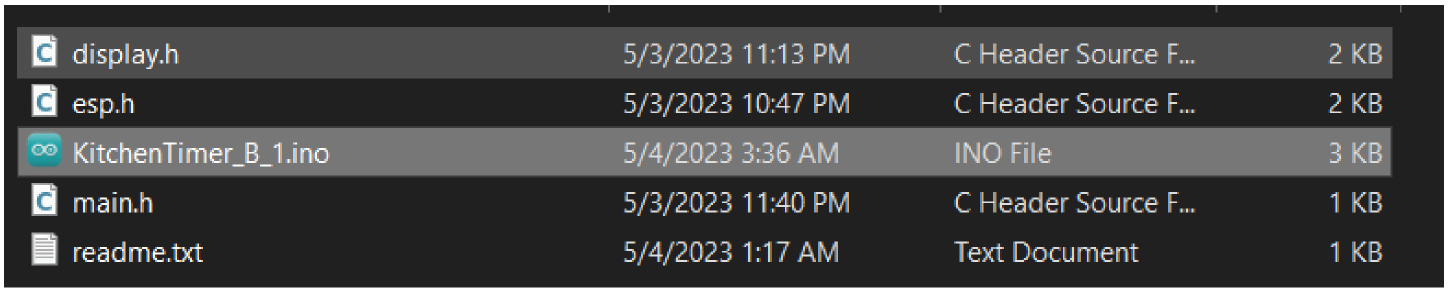
5. Assemble Stage

This section is not applicable at the moment as the board and enclosure have not been assembled or tested.

6. Software Development

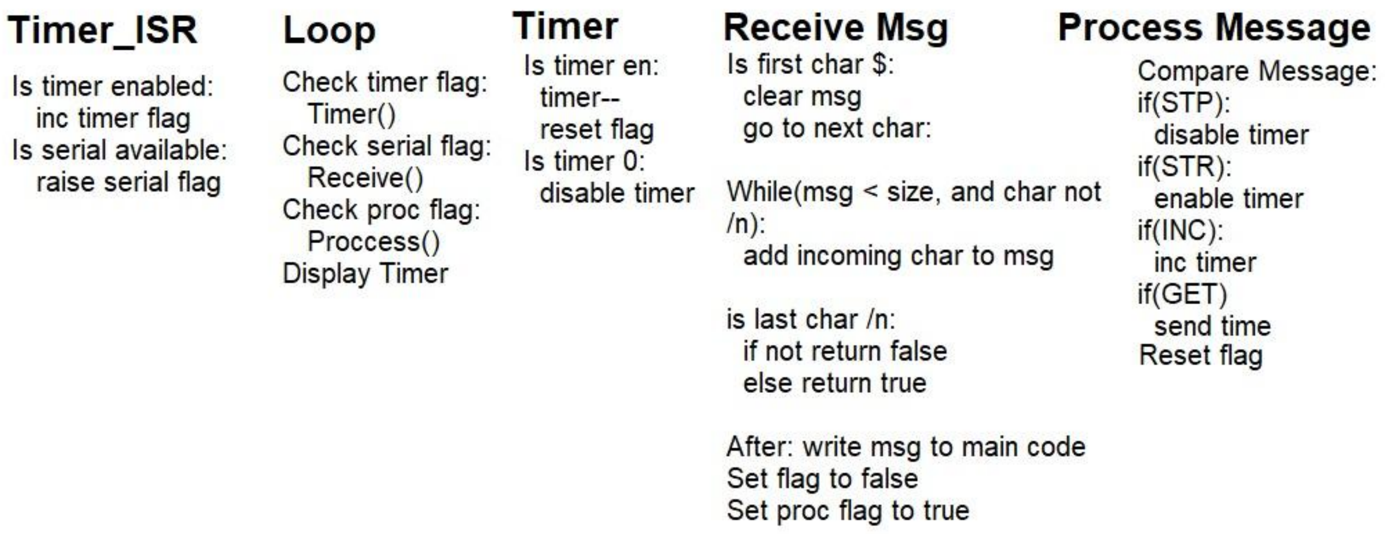
The code is set up using a modular style that obeys the Mutex - Embedded C Style. It uses separate header files for each peripheral along with a main header which sets up interrupts and will contain more simple peripherals (such as the buzzer or buttons). The ino file uses interrupt polling to communicate with the ESP. The interrupt occurs every .4 ms and checks for incoming data from the ESP. Additionally the interrupt increments a flag to change to a timer. The main loop has four tasks. First it checks the timer interrupt to see if a second has passed, if so the timer decrements. The second checks to see if there is a message incoming. As the process function uses a while loop, this flag will only be raised when the controller thinks the entire message is ready to be received. This shouldn't be a problem as the message is 3 chars long(excluding the $ and \n.) If the message is successfully processed a flag will be set to process the message. The process function compares the message and performs the appropriate task. At the end of the loop the timer value is displayed. The code can be viewed in \Arduino_Code\KitchenTimer_B_1\

Proccess



display.h	5/3/2023 11:13 PM	C Header Source F...	2 KB
esp.h	5/3/2023 10:47 PM	C Header Source F...	2 KB
KitchenTimer_B_1.ino	5/4/2023 3:36 AM	INO File	3 KB
main.h	5/3/2023 11:40 PM	C Header Source F...	1 KB
readme.txt	5/4/2023 1:17 AM	Text Document	1 KB

Figure 4. Setup of the code as previously described.



```
Timer_ISR
Is timer enabled:
   inc timer flag
Is serial available:
   raise serial flag

Loop
Check timer flag:
   Timer()
Check serial flag:
   Receive()
Check proc flag:
   Proccess()
Display Timer

Timer
Is timer en:
  timer--
  reset flag
Is timer 0:
  disable timer

Receive Msg
Is first char $:
  clear msg
  go to next char:

While(msg < size, and char not
/n):
  add incoming char to msg

is last char /n:
  if not return false
  else return true

After: write msg to main code
Set flag to false
Set proc flag to true

Process Message
Compare Message:
if(STP):
  disable timer
if(STR):
  enable timer
if(INC):
  inc timer
if(GET)
  send time
Reset flag
```

Figure 5. Code diagram. Note that the serial flag is only raised when avail > 4 (expected avail = 5) to prevent the while loop from hanging.

7. Enclosure Design

The enclosure is very simple. It is a half shell for the PCB. This was chosen as a proper full shell would need some form of risers as the buttons are shorter than the display and buzzer. If needed it is possible to make a simple friction fitting top for the current design. The shell has 4 cylinders/pins which line up with the mounting holes of the PCB. The shell was designed so that the PCB fits perfectly. This means once assembled it is very likely that the shell or PCB itself will need to be sanded to ensure a proper fit. Note that this enclosure has not yet been printed. Despite this the print is expected to be very simple.

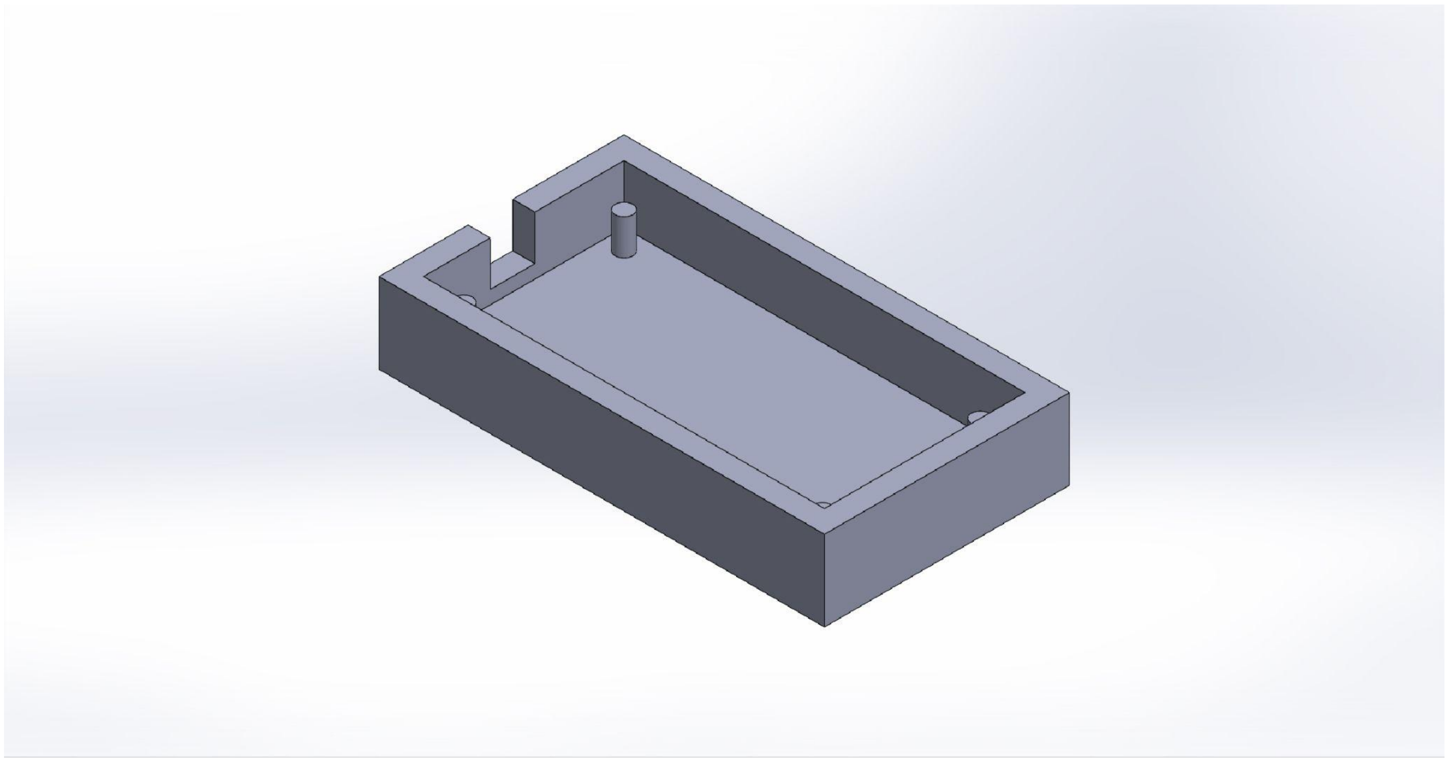


Figure X. Render of the enclosure.