Contactless Thermometer System Verification Documentation

By: Tyrone Stagner, Jordan Hendricks, Jimmy Parra, and Caspian Hedlund

Date Submitted: May 29, 2021

Project Mentor: Karthik Gopalakrishnan

Contents

1 Block Diagram and Interface Definitions

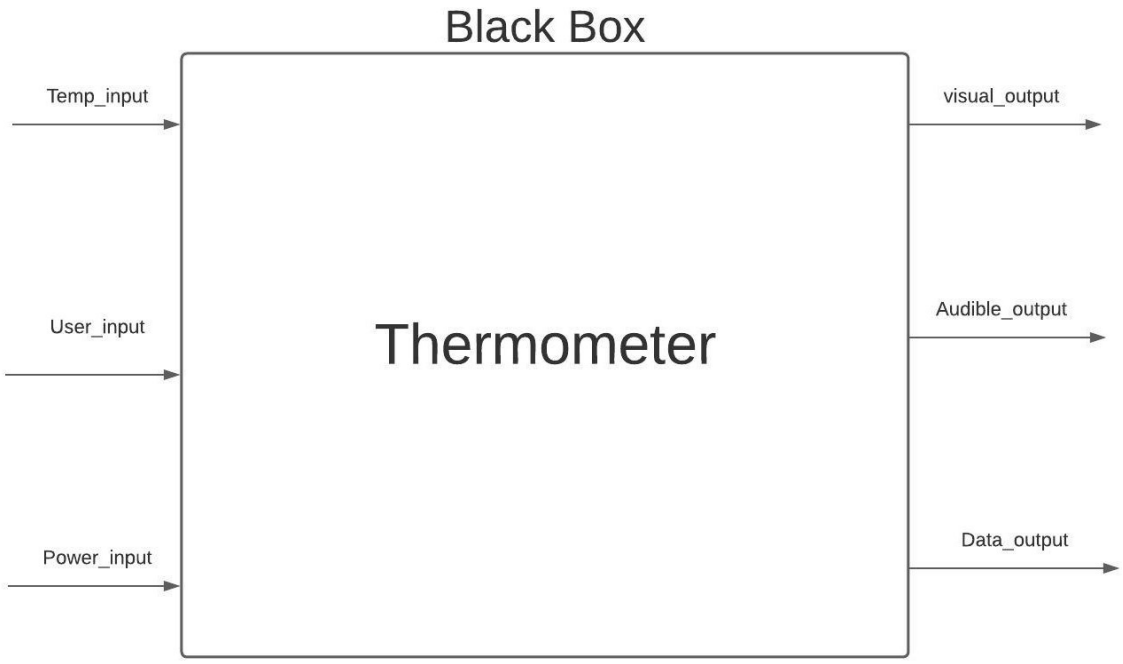

Figure 1: Black Box Diagram

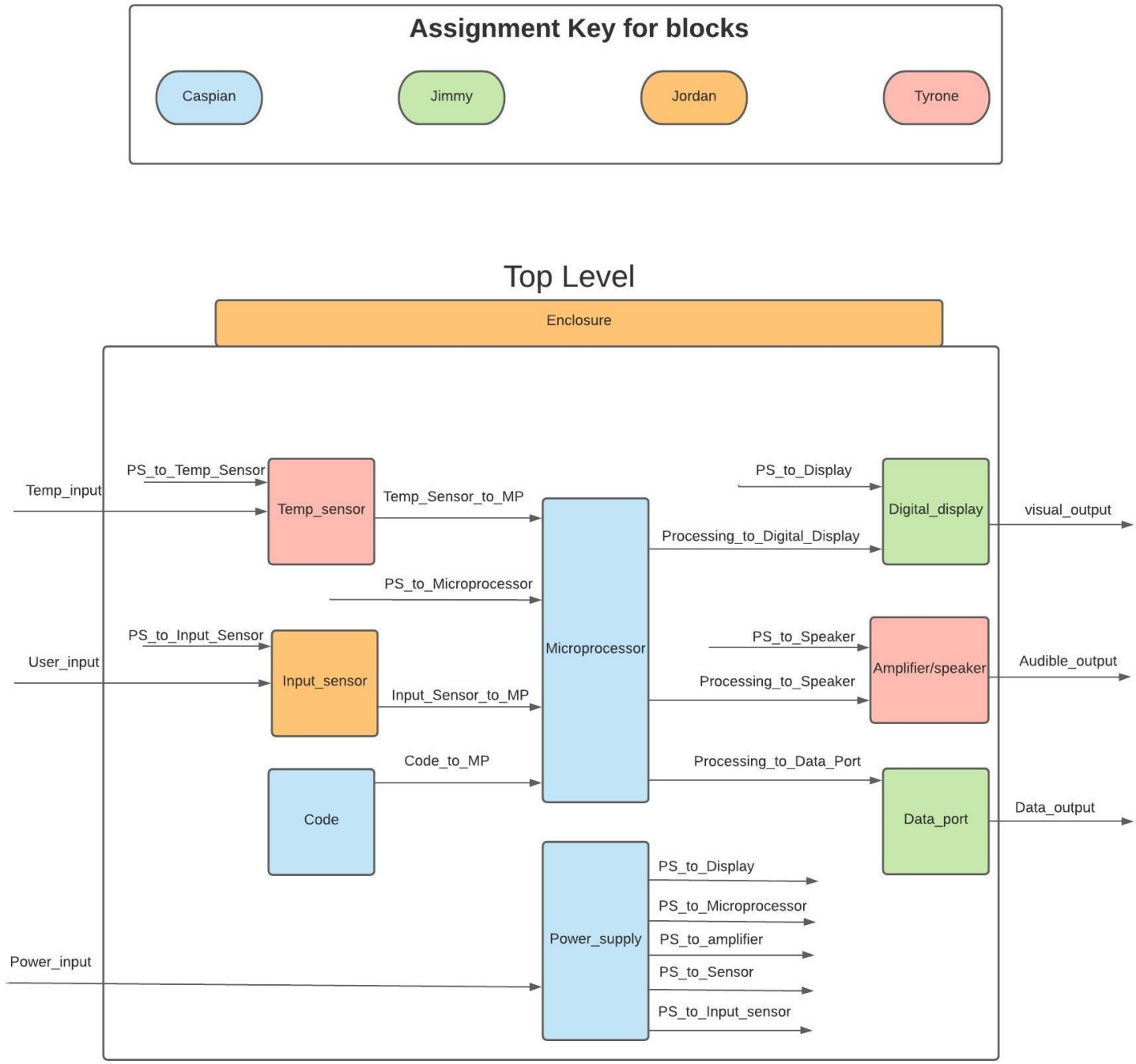

Figure 2: Top Level Block Diagram and Block Authors

Name	Type	Detail
Temp_input	Temperature	Range: 90-110 °F, accuray within 1 °F, at least 1 cm distance from sensor to take temp, no contact.
User_input	Binary	Distinguish between user, intuitive
Power_in	DC	5 Volt, 3 Amp
Visual_output	LCD or OLED	Displays temp, uses I2C pins to communicate.
Audible_output	Audio output	Beeps once when temperature scan is complete, beeps twice if fever is detected.
Data_output	Flash memory	Data, .text file
PS_to_Temp_Sensor	DC	3.3V, 25 mA
PS_to_Input_Sensor	DC	3.3V, 10 mA
PS_to_Microprocessor	DC	Built in to Raspberry Pi Zero
PS_to_Display	DC	3.3V, 2.5 mA
PS_to_Speaker	AC	3.3V, 500 mA
Temp_Sensor_to_MP	I2C interface	Verify accuracy is within 1 °F, verify it is able to measure a temperature range of 90-110 °F.
Input_Sensor_to_MP	DC	Verify correct user selected, 3.3V.
Code_to_MP	Python	Verify correct input is detected from input and temperature sensor, verify correct output is sent to display, data port, and speaker.
Processing_Data_to_Digital_Display	I2C interface	5V, 21.1 mA. Verify correct text is displayed for healthy temperatures and fever.
Processing_Data_to_Speaker	AC	5V. Verify correct sound is played for healthy temperatures and fever.
Processing_Data_to_Data_Port	TX, RX	Verify temperatures stored to document are correct, verify the document is correctly formatted.

Table 1: Interface Definitions

2 Electrical Schematic and PCB Trace

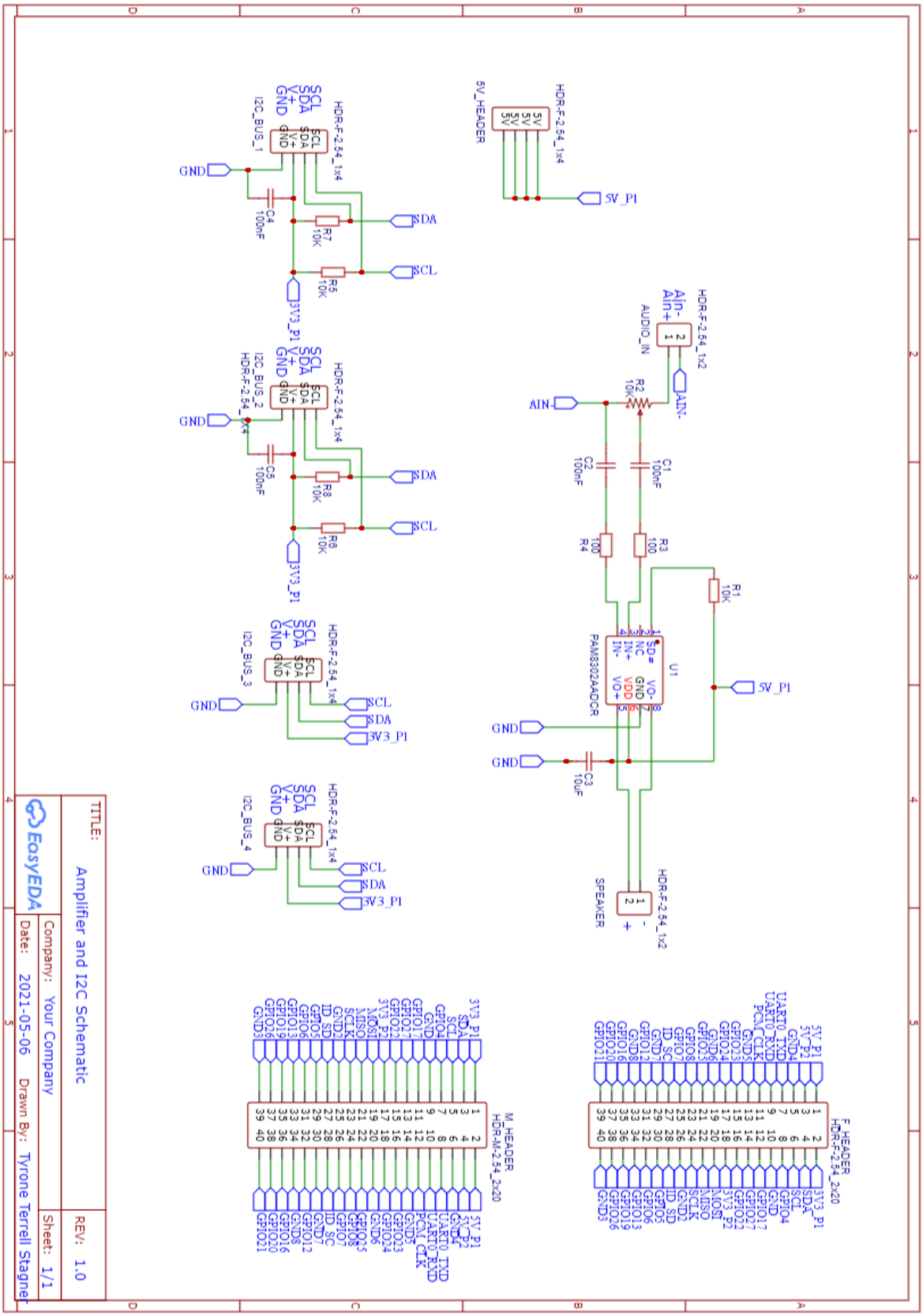

Figure 3: Electrical Schematic

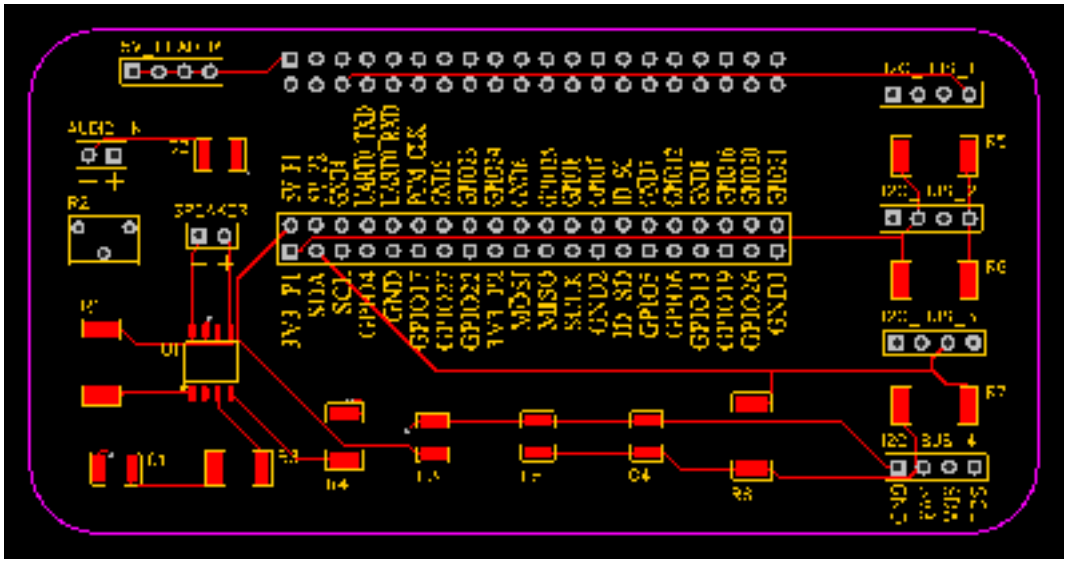

Figure 4: PCB trace

3 Mechanical Drawings

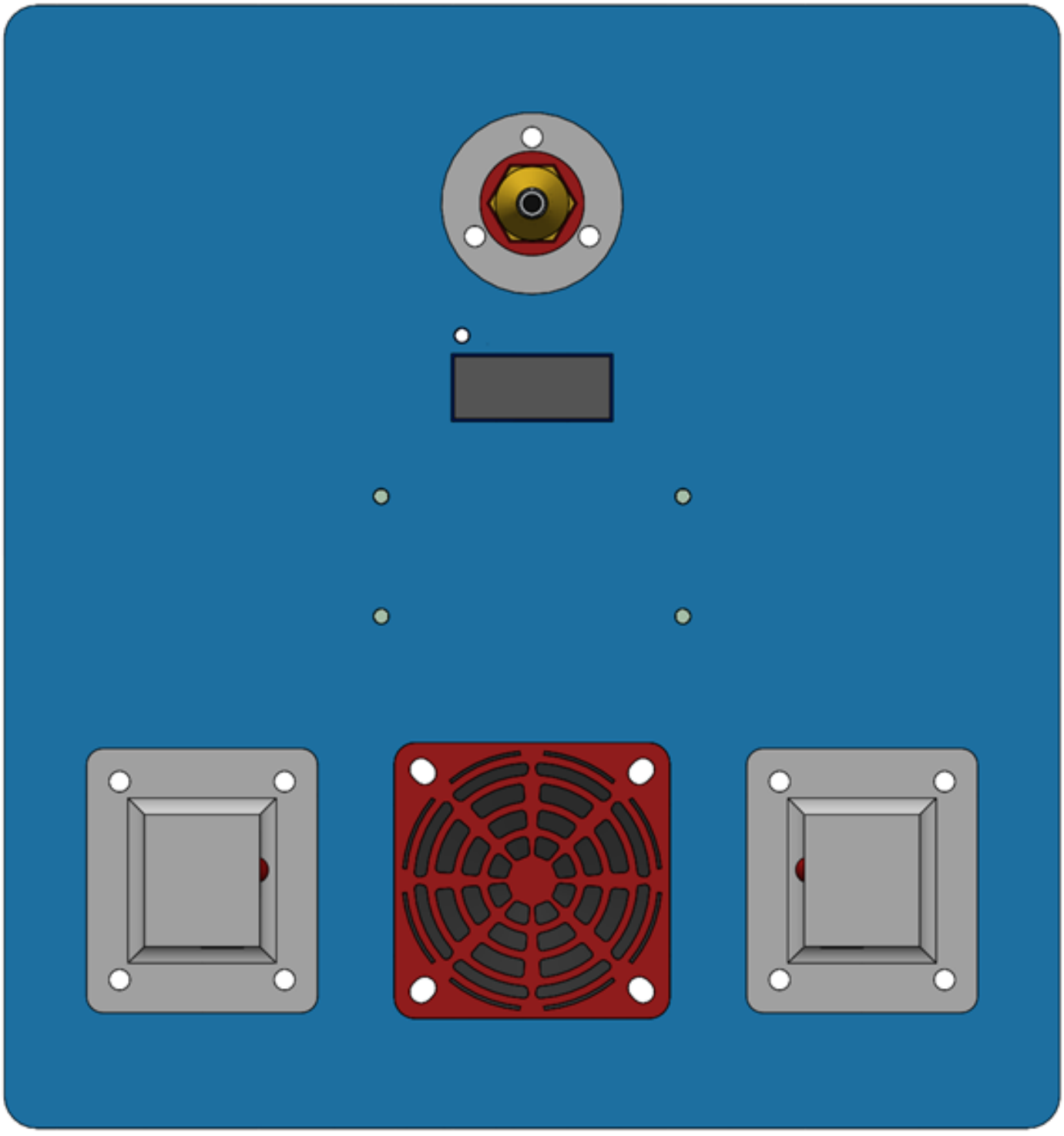

Figure 5: Rendered front view of the device

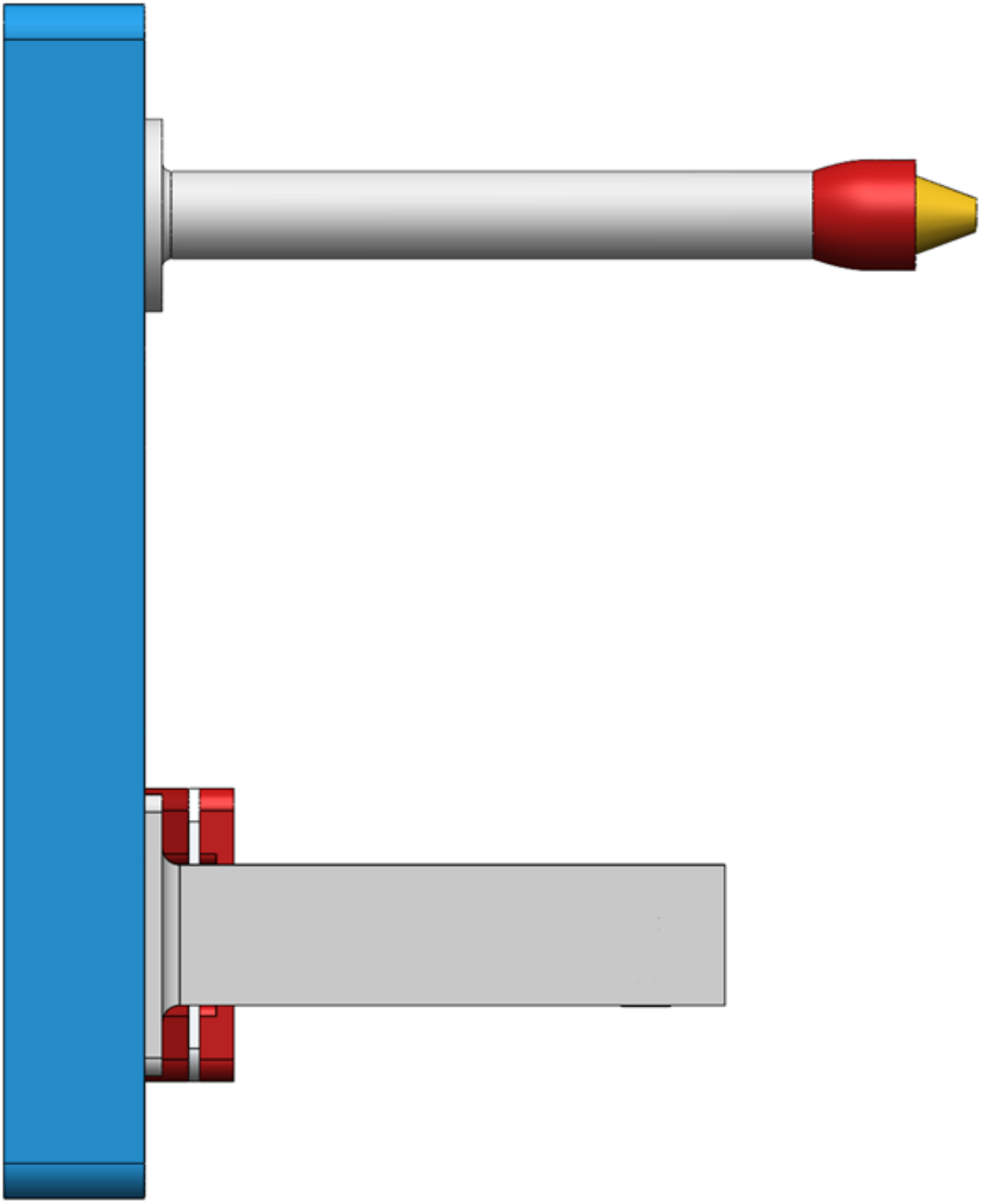

Figure 6: Rendered side view of the device

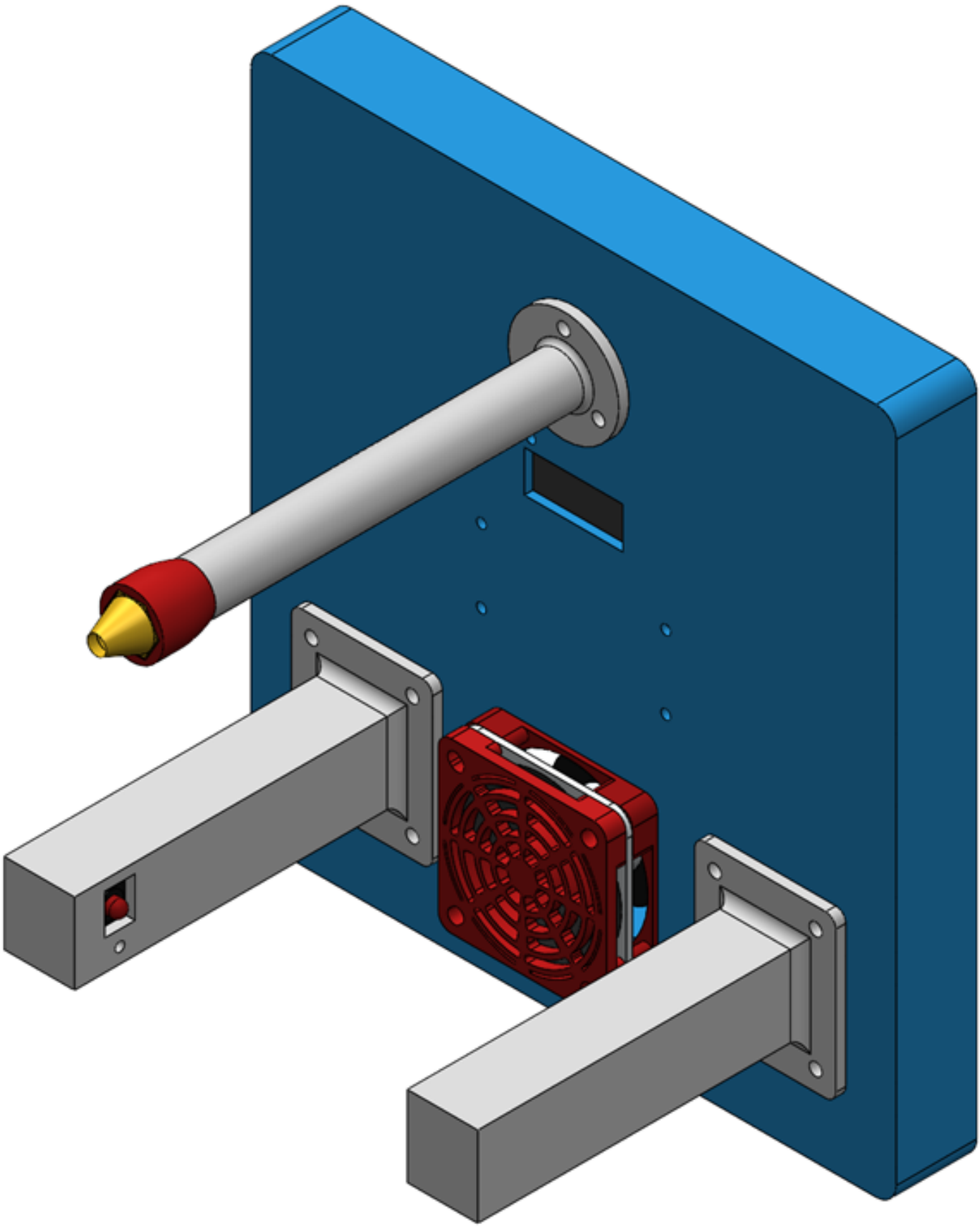

Figure 7: Rendered front perspective view of the device

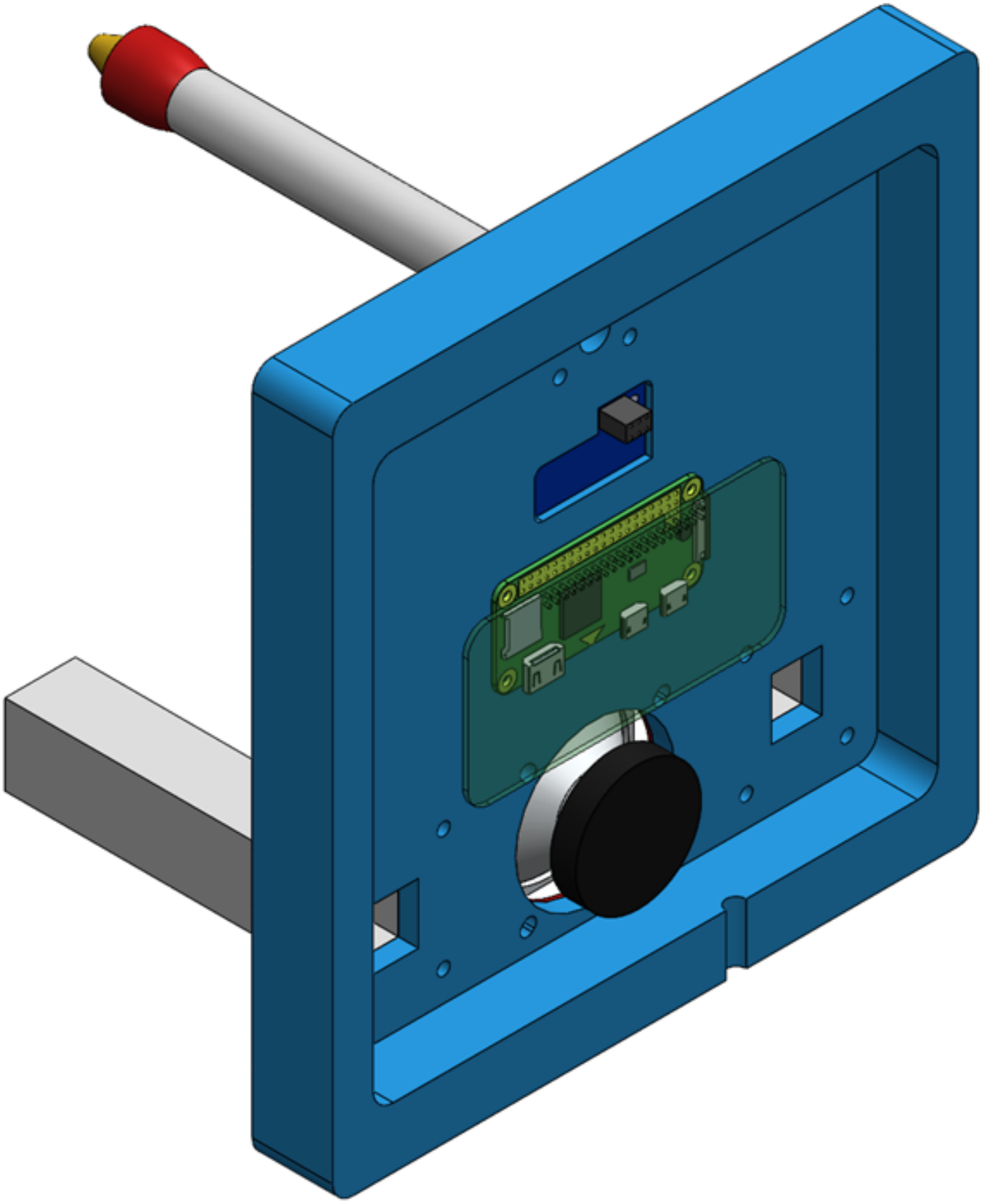

Figure 8: Rendered rear perspective view of the device

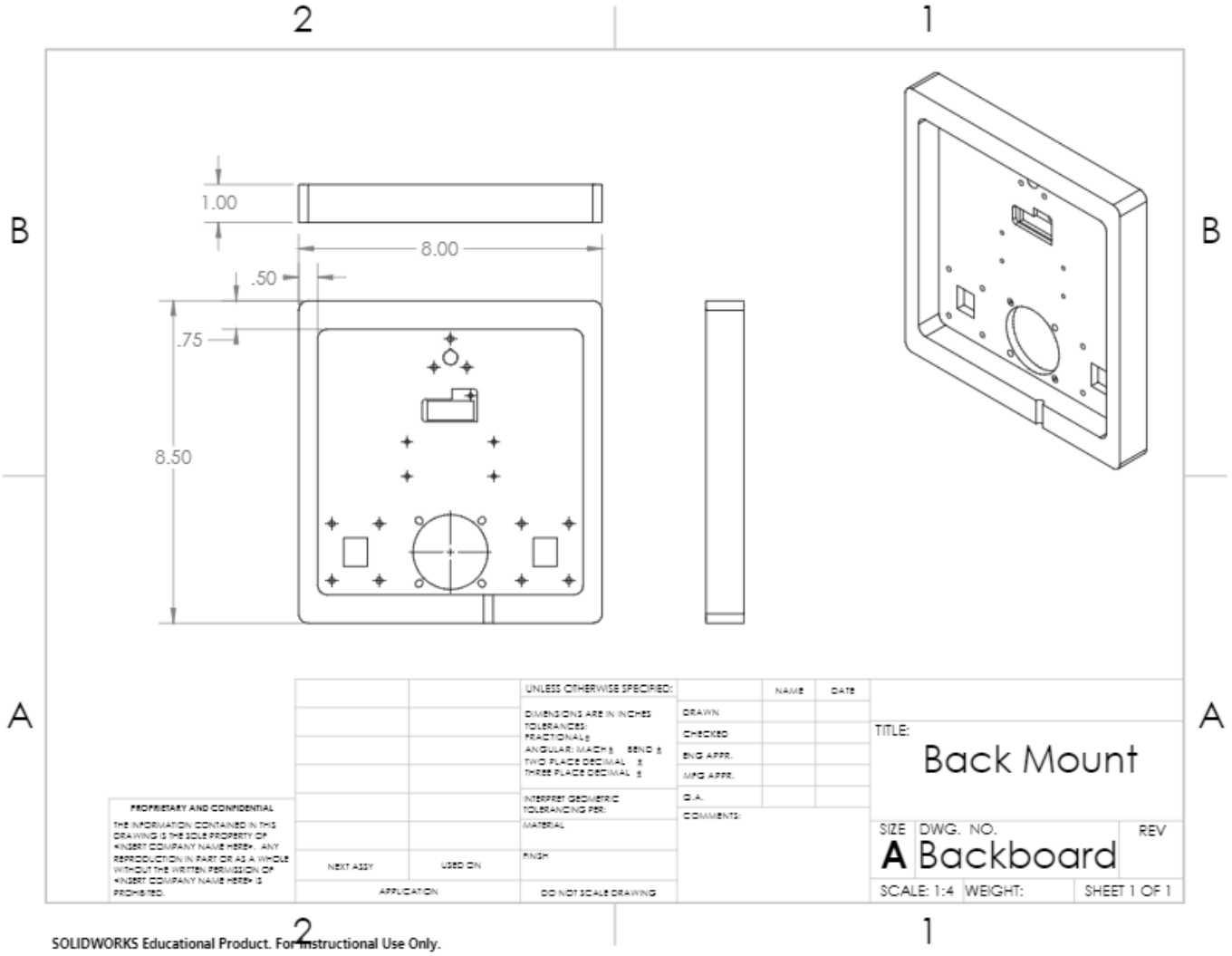

Figure 9: Backboard mechanical drawing

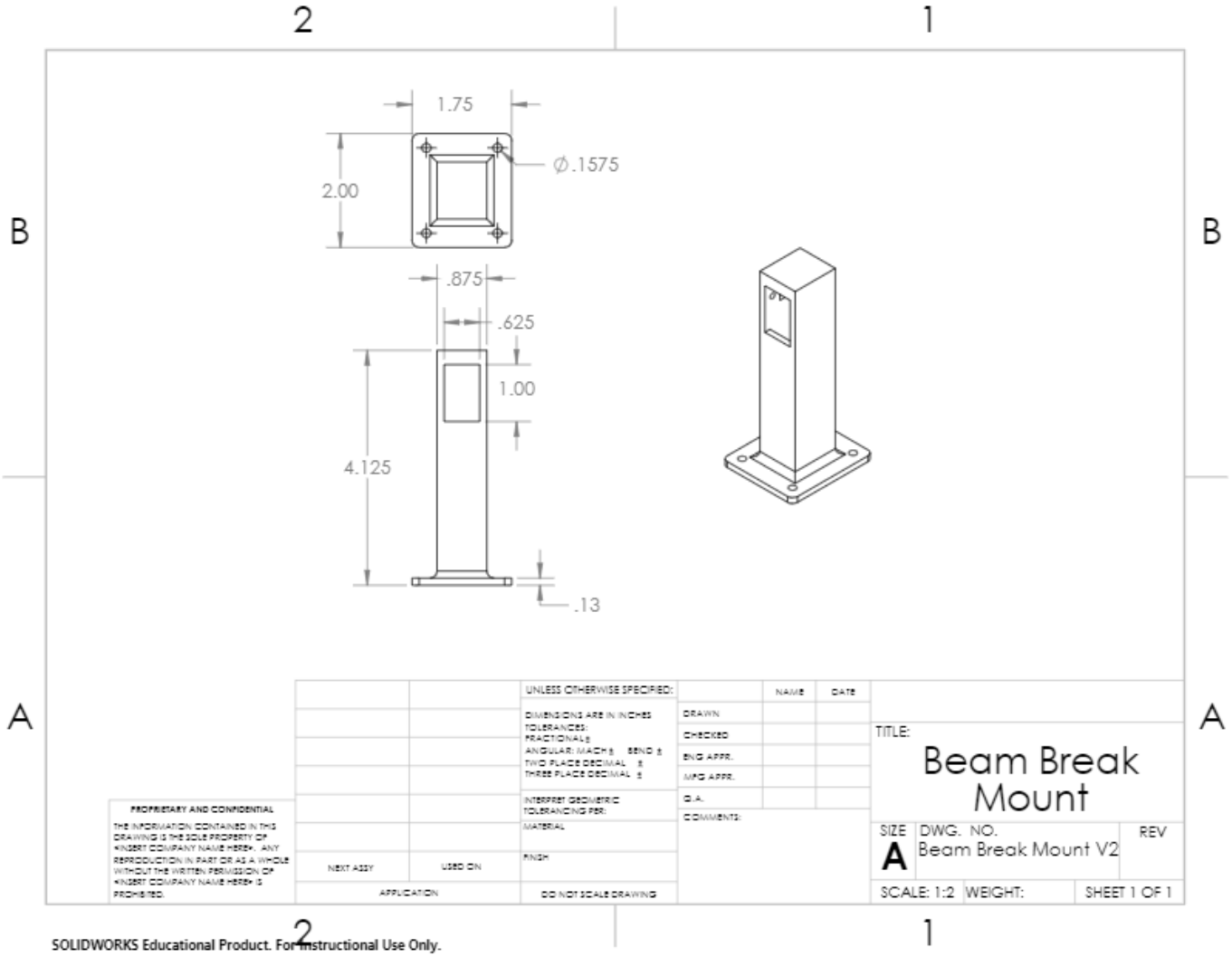

Figure 10: Beam break sensor mount mechanical drawing

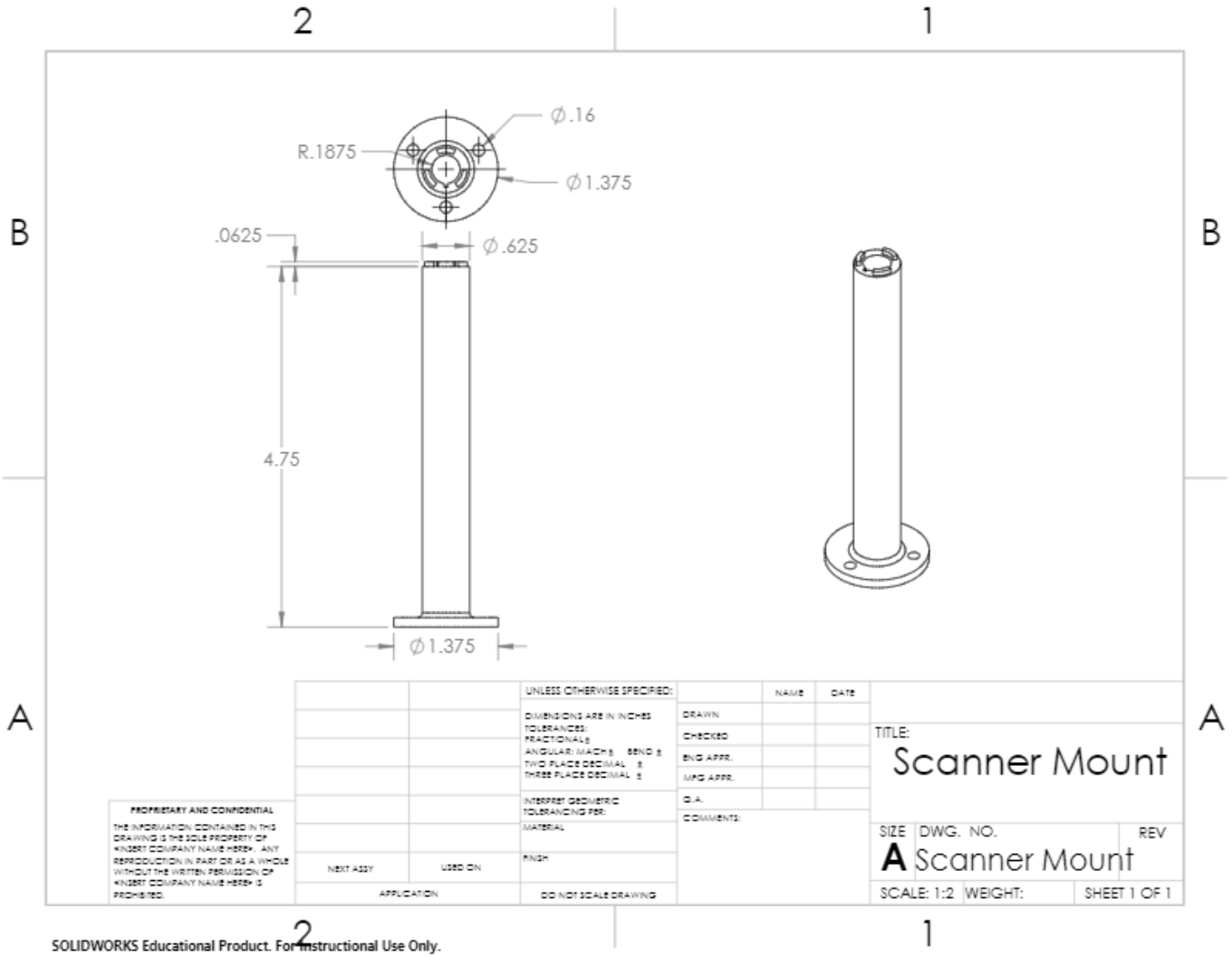

Figure 11: Temperature sensor mount base mechanical drawing

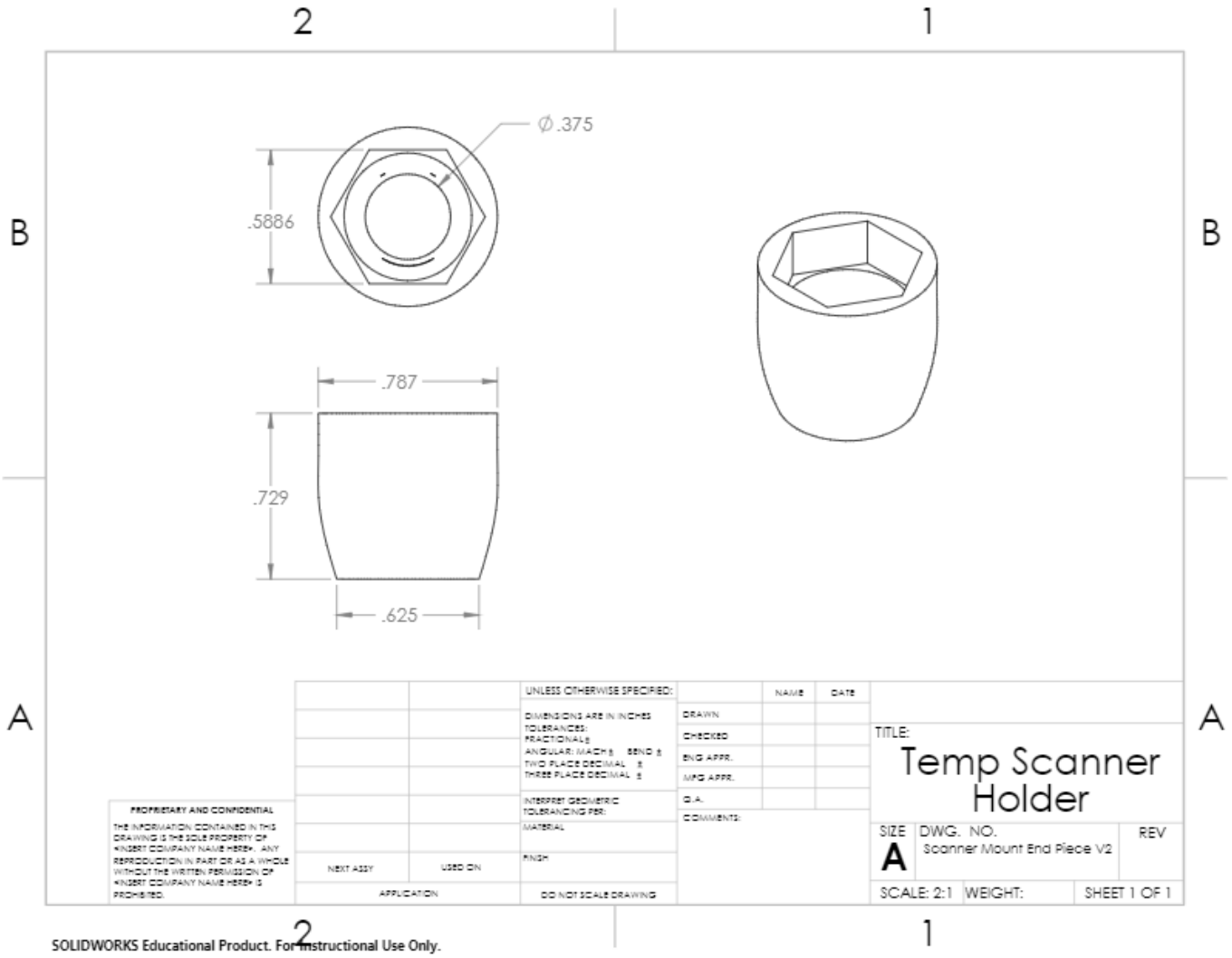

Figure 12: Temperature sensor mount end mechanical drawing

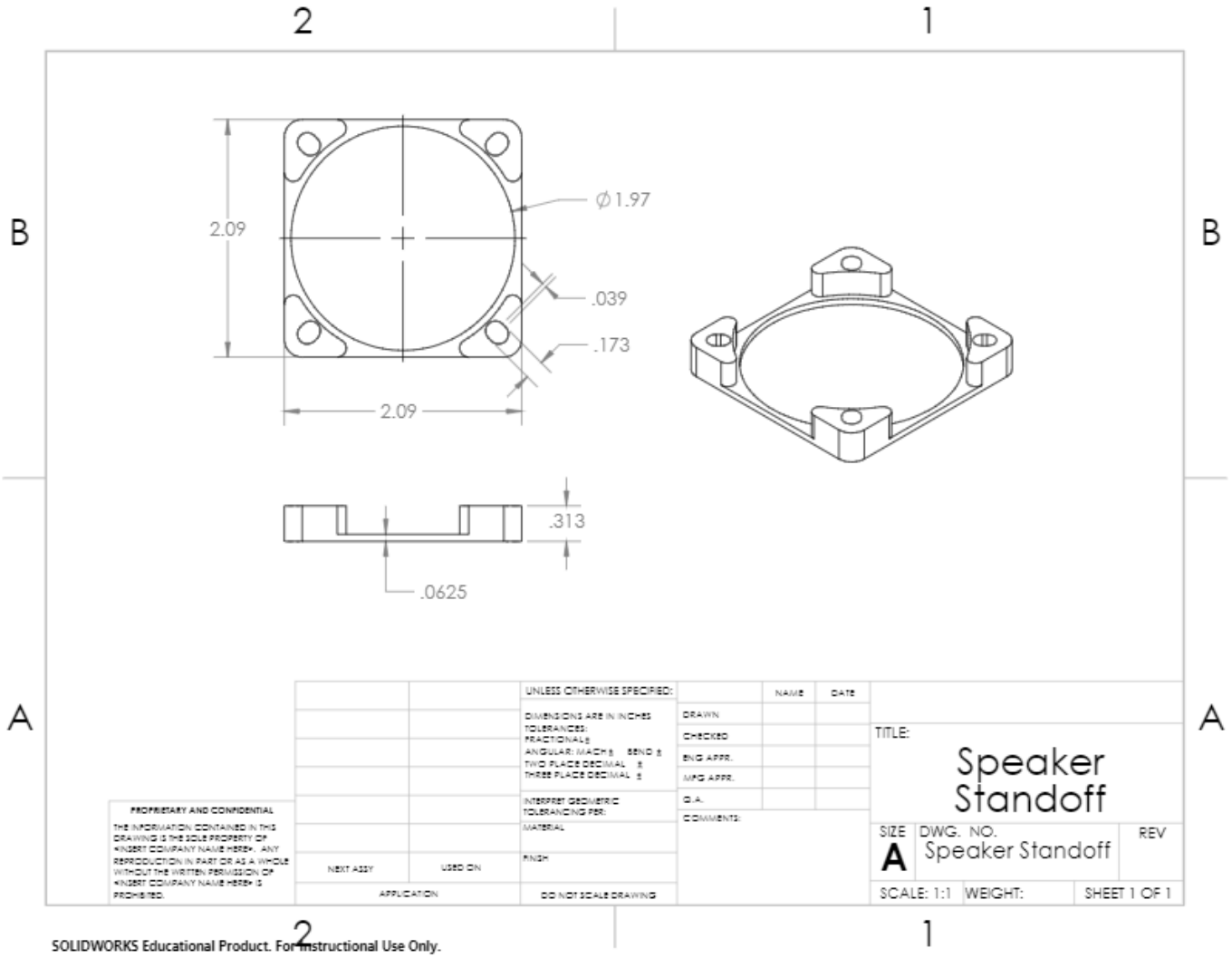

Figure 13: Speaker mount mechanical drawing

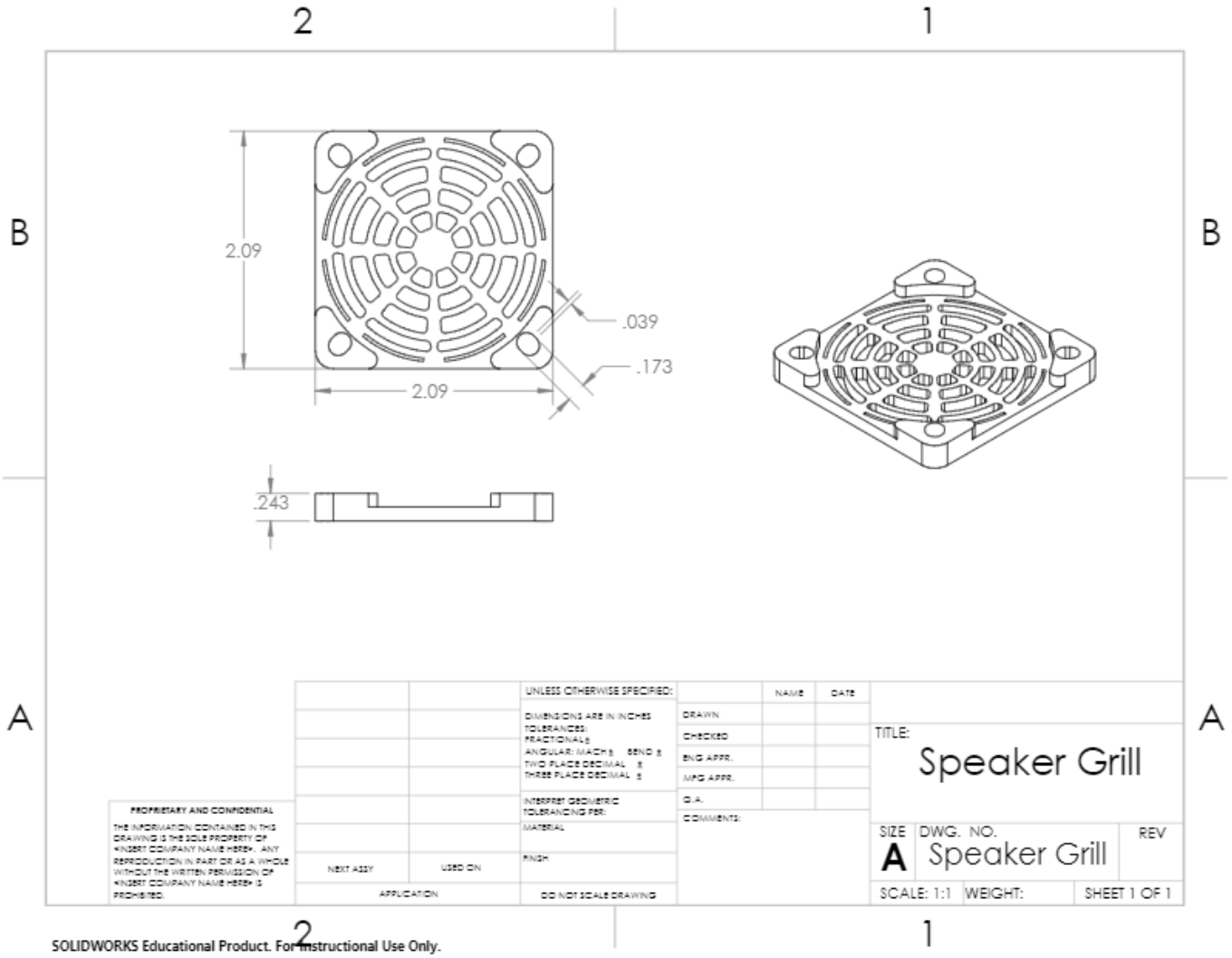

Figure 14: Speaker grill mechanical drawing

4 Time Report

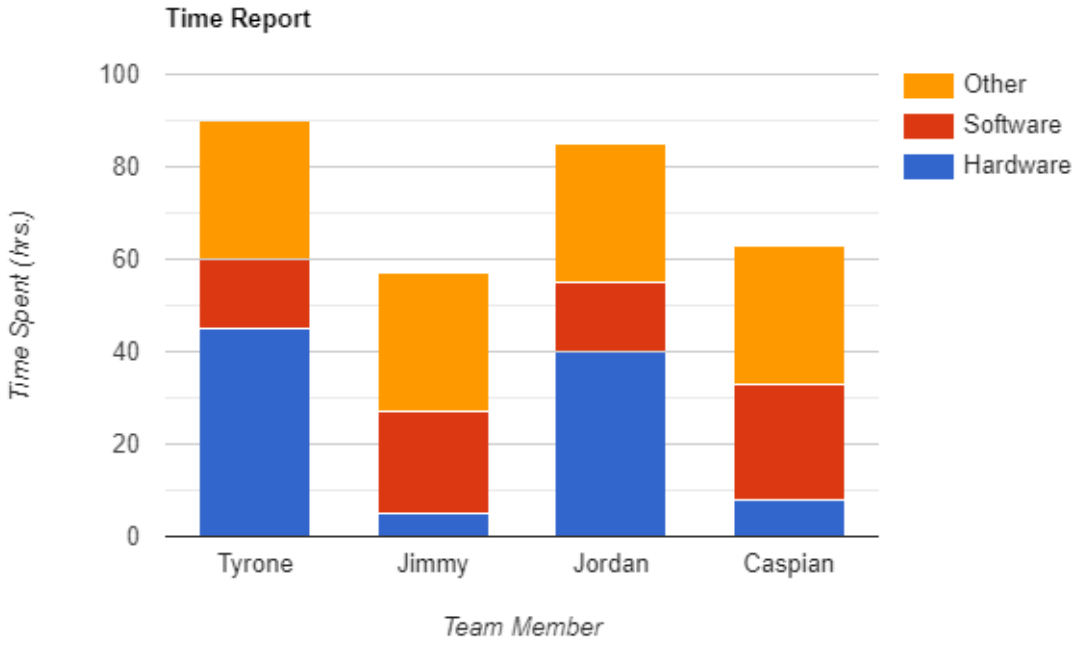

Figure 15: Time report

5 Bill of Materials

Electronics BOM							
Quantity	Unit	Reference Designators	Partname	Part Number	Cost	URL to part	URL to Data sheet
1	each	Microprocessor	Raspberry Pi Zero W	3708	$14.00	https://www.adafru	https://www.raspberr
1	each	Digital_Display	128x32 Mini OLED Display	3527	$14.95	https://www.adafru	https://cdn-learn.ada
1	each	Temp_sensor	Contact-less Infrared Scanner	MLX90614	$15.95	https://www.adafru	https://cdn-shop.ada
1	each	Input_sensor	IR Beam Break Sensor	2167	$1.95	adafruit.com/produ	https://media.digikey
2	each	AUDIO_IN, SPEAKER	HDR-F-2.54_1x2	C49661	$0.03	https://lcsc.com/pr	https://datasheet.lcs
4	each	C1,C2,C4,C5	100nF	C527375	$0.12	https://lcsc.com/pr	https://datasheet.lcs
1	each	C3	10uF	C380537	$0.25	https://lcsc.com/pr	https://datasheet.lcs
1	each	F_HEADER	HDR-F-2.54_2x20	C50982	$0.16	https://lcsc.com/pr	https://datasheet.lcs
5	each	5V_HEADER, I2C_BUS_1, I2C_BUS_2, I2C_BUS_3, I2C_BUS_4	HDR-F-2.54_1x4	C225501	$0.09	https://lcsc.com/pr	https://datasheet.lcs
1	each	M_HEADER	HDR-M-2.54_2x20	C50980	$0.12	https://lcsc.com/pr	https://datasheet.lcs
5	each	R1, R5, R6, R7, R8	10K	C270956	$0.03	https://lcsc.com/pr	https://datasheet.lcs
1	each	R2	10K	C81167	$0.32	https://lcsc.com/pr	https://datasheet.lcs
2	each	R3,R4	100	C63036	$0.04	https://lcsc.com/pr	https://datasheet.lcs
1	each	U1	PAM8302AADCR	C112137	$0.23	https://lcsc.com/pr	https://datasheet.lcs
1	each	JLCPCB	PCB	Y3-3565453A	$20.00		
1	each	N/A	SAMSUNG EVO Select 64GB microSDXC	MB-ME64HA	$10.99	http://www.amazo	
				Subtotal:	$79.23		

Enclosure BOM				
Quantity	Size	Part in Assembly	Length	Estimated Cost
3	M4	Temperature Scanner Mount to Backboard	12-15mm	$1
8	M4	Beam Break Sensor Mount to Backboard	13-15mm	$3
4	M3	Raspberry Pi Mount to Backboard	10-13mm	$1
1	M3	OLED Mount to Backboard	8-10mm	$1
4	M4	Speaker Mount to Backboard	25-27mm	$1
2	M3	Beam Break Sensor Mount	13-15mm	$1
4		Command Strips		$3
1 Kg		3D Printer Fillament for Enclosure		$25
			Subtotal:	$36
			Total:	$115

Table 2: Bill of materials

6 Python Code

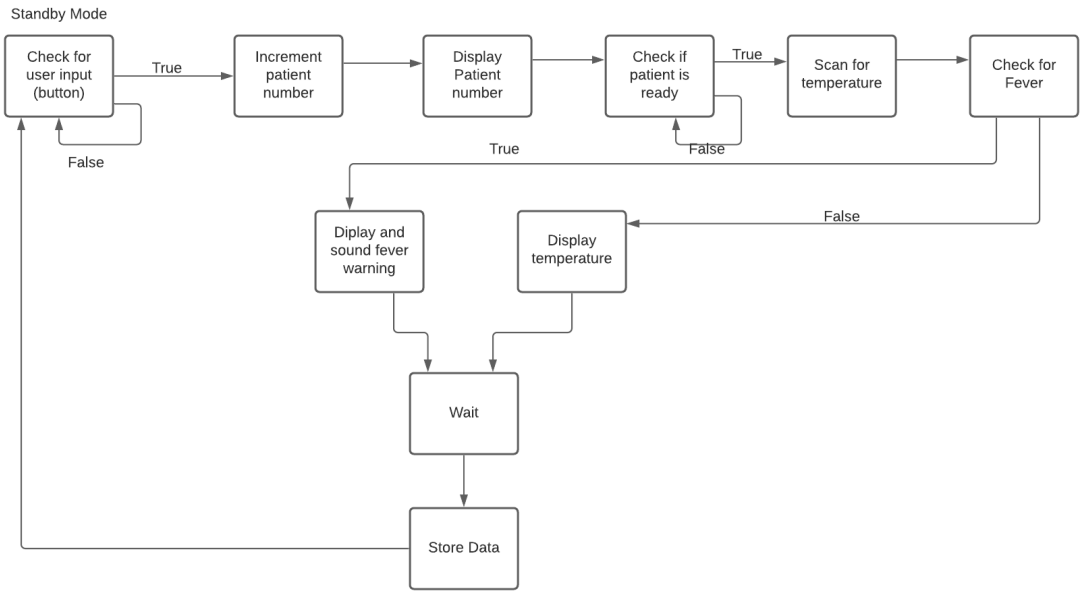

Figure 16: Main program code flow chart

1. MAIN1.py

```
# The following people helped create the code below
# Caspian Hedlund, Jimmy Parra, Jordan Hendricks, Tyrone Stagner
# File modified on 5/27/2021

from gpiozero import LED, Button
from signal import pause
from mlxfirmware import MLXSERIES
import time
import os
import ThermometerFunctions
from DisplayFuncs import *
from UserLogging import *


userNum = 1
curTemp = 0
mainLooper = 1
ledg = ['time', 'user', 'temp']
rows = []

MLX_IR_ADD = 0x5a #define I2C register address

#Pass register address back into MLXSERIES class and store read value
therm_data = MLXSERIES(MLX_IR_ADD)

#Main Program
InitDisp() #Initialize the OLED display
while mainLooper == 1:

    # will set gpio alt pins for 13 to make
```

```
    # sure speaker will work every time.
    os.system('gpio_alt -p 13 -f 0')

    ReadyDisp() # Will disply when ready

    #loop until input is detected
    while ThermometerFunctions.checkForInput() == False:
        pass #Do nothing until the user wakes the system up

    clearDisp() #clears display
    UserDisp(userNum) #Display current user number

    # will call omxplayer and play wav file
    os.system('omxplayer /home/pi/Documents/Place.wav')

    #Calls ThermometerFunctions and creates a beep
    ThermometerFunctions.beep()

    clearDisp() #Clear Display

    curTemp = therm_data.get_TEMP() #Get temperature

    TempDisp(curTemp) #Display temperature
    time.sleep(4)
    # If power is lost it will restart the user
    #number to 1, but displays time and date for each user.
    # you will be able to tell when power is lost
    #by it reseting the user number to 1.

    getInfo(rows, userNum, curTemp) #gets info
    CreateCsv(ledg, rows) # Creates the Text dat for CSV file
    userNum = userNum +1 # will increment the user by 1 each time

    if float(curTemp) >= 100.4:
        # will set gpio alt pins for 13 to make sure speaker will work every time.
        os.system('gpio_alt -p 13 -f 0')
        os.system('omxplayer /home/pi/Documents/Fever.wav')

    time.sleep(1) # will sleep for  4 seconds before clearing diplay
    clearDisp() #Turn off display
```

2. ThermometerFunctions.py

```
# The following people helped create the code below
# Caspian Hedlund, Jimmy Parra, Jordan Hendricks, Tyrone Stagner
# File modified on 5/27/2021

from gpiozero import LED, Button, TonalBuzzer
from gpiozero.tones import Tone
import time


#Functions

#Checks if the beam on the beam break sensor is broken
def checkForInput():
    btn = Button(17)
    btn.wait_for_press()
    time.sleep(1)
```

```
    return True

#Beep once
def beep():
    speaker = TonalBuzzer(13) #Set up speaker GPIO pin
     #Beep a 550 Hz tone
    speaker.play(Tone(550))
    time.sleep(0.25)
    speaker.stop
```

3. UserLogging.py

```
# The following people helped create the code below
# Caspian Hedlund, Jimmy Parra, Jordan Hendricks, Tyrone Stagner
# File modified on 5/27/2021
import csv
from datetime import datetime
import random

# Store the given information into a dynamic array
def getInfo(rows, UserNum, Temp):
  now = datetime.today()
  a = [now.strftime("%m/%d/%Y %H:%M:%S"), "User Number: "  + str(UserNum),
  str(Temp)]
  rows.append(a)

# Creates the CSV file from the information from the array
def CreateCsv(Legend, rows):
  with open('User_Data', 'a+') as f:

# using csv.writer method from CSV package
    write = csv.writer(f)

    write.writerow(Legend)
    write.writerows(rows)
```

4. DisplayFuncs.py

```
# The following people helped create the code below
# Caspian Hedlund, Jimmy Parra, Jordan Hendricks, Tyrone Stagner
# File modified on 5/27/2021

import time
import subprocess

# SPDX-FileCopyrightText: 2017 Tony DiCola for Adafruit Industries
# SPDX-FileCopyrightText: 2017 James DeVito for Adafruit Industries
# SPDX-License-Identifier: MIT
from board import SCL, SDA
import busio
from PIL import Image, ImageDraw, ImageFont
import adafruit_ssd1306

# Create i2c interface
i2c = busio.I2C(SCL, SDA)

# Create the SSD1306 OLED class.
# The first two parameters are the pixel width and
# pixel height.  Change these
# to the right size for your display!
```

```
disp = adafruit_ssd1306.SSD1306_I2C(128, 32, i2c)

# Create blank image for drawing.
# Make sure to create image with mode '1' for 1-bit color.
width = disp.width
height = disp.height
image = Image.new("1", (width, height))

# Get drawing object to draw on image.
draw = ImageDraw.Draw(image)
font = ImageFont.truetype("DejaVuSans.ttf", 12)

# Initilize the display
def InitDisp():
  # Draw a black filled box to clear the image.
  draw.rectangle((0, 0, width, height), outline=0, fill=0)

  # Draw some shapes.
  # First define some constants to
  # allow easy resizing of shapes.
  padding = -2
  top = padding
  bottom = height - padding


  draw.rectangle((0, 0, width, height), outline=0, fill=0)

# Clear the display
def clearDisp():
  disp.fill(0)
  disp.show()
  draw.rectangle((0, 0, width, height), outline=0, fill=0)

# Display the current temperature and if a fever was detected
def TempDisp(curTemp):
  draw.text((0, 0), "   Temperature: \n" + str(curTemp),
  font=font, fill=255, align = "center")
  disp.image(image)
  disp.show()
  time.sleep(3)
  clearDisp()
  if curTemp >= 100.4:
    draw.text((0, 20), "You have a fever!! \n",
    font=font, fill=255, align = "center")
  disp.image(image)
  disp.show()

# Display the current user number
def UserDisp(UserNum):
  draw.text((0, 0), " User Number: \n " + str(UserNum),
  font=font, fill=255, align = "center")
  disp.image(image)
  disp.show()
# Display when ready to take temp
def ReadyDisp():
  draw.text((0, 0), " Ready \n " +"to take temp " ,
  font=font, fill=255, align = "center")
  disp.image(image)
```

```
    disp.show()
```

5. mlxfirmware.py

```
"""
The following people helped create the code below
Caspian Hedlund, Jimmy Parra, Jordan Hendricks, Tyrone Stagner
File modified on 5/27/2021

Reading the data sheet and looking at some code from adafruit
that is programmed in C for an arduino and looking at the
code from a user on github I was able to get this to work
finally. The git hub web page is
"https://github.com/CRImier/python-MLX90614/blob/master/mlx90614.py"

The data sheet says we are able to acces diffrent registers
by diffrent hex values to get the data from the MLX 60914.
We are only going to need to use the hex values 0x07 which
points to the correct address for getting the tempature.
We are also able to use the Hex value 0x06 which will display
the tempature of the sensor or ambiant tempature.
"""
#we need to import SMBUS to get it to work on the pi
import smbus
#We are going to import time and sleep so we are able to
#use them int he code when needed.
import time

#We need to create a class so that we can import it inot
#the main progrma for the project.
class MLXSERIES():

#We are creating varibles to have them equal the hex values
#that we are accessing in the sensor.
    MLX90614_AMBTEMP=0x06
    MLX90614_TEMP=0x07

    comattempts = 5
#have to make the connection to the smbus  and define the
#address we are goning to be  reading data from
# has to be __init__ becuase it will throw a constructor error.
    def __init__(self, address=0x5a, busport=1):
        self.busport = busport
        self.address = address
        self.bus = smbus.SMBus(bus=busport)

#This is going to go through and try and make a connection to the
#register in the device if it can not detet it will throw in error
    def read_reg(self, reg_addr):
        err = None
        for i in range(self.comattempts):
            try:
                return self.bus.read_word_data(self.address, reg_addr)
            except IOError as e:
                err = e
                #"Rate limiting" - sleeping to prevent problems with sensor
                #when requesting data too quickly
                sleep(0.1)
```

```
            #By this time, we made a couple requests and the sensor didn't respond
            #(judging by the fact we haven't returned from this function yet)
            #So let's just re-raise the last IOError we got
            raise err

    #we need to define the data that we are getting and perform
    #the calculation to get the temp. The 0.02 comes from the
    #data sheet telling us to do this to calculate the correct temp.
        def data_to_temp(self, data):
            temp = (((data*0.02) - 273.15)*(9/5)) + 35
            temp = round(temp,1)
            return temp

    #we need to get the data for the AMBTEMP  register and then pass it
    #into the data_to_temp to calculate the tempature
        def get_AMBTEMP(self):
            data = self.read_reg(self.MLX90614_AMBTEMP)
            return self.data_to_temp(data)

    #we need to get the data for the TEMP register and then pass it into
    #the data_to_temp to calculate the tempature
        def get_TEMP(self):
            data = self.read_reg(self.MLX90614_TEMP)
            return self.data_to_temp(data)

    # this just helps us print the tempature of AMBTEMP and TEMP, we will have to use
    # these fuctions in our main program to get the tempatures for them.
    if __name__ == "__main__":
        sensor = MLXSERIES()
        print(sensor.get_AMBTEMP())
        print(sensor.get_TEMP())
```