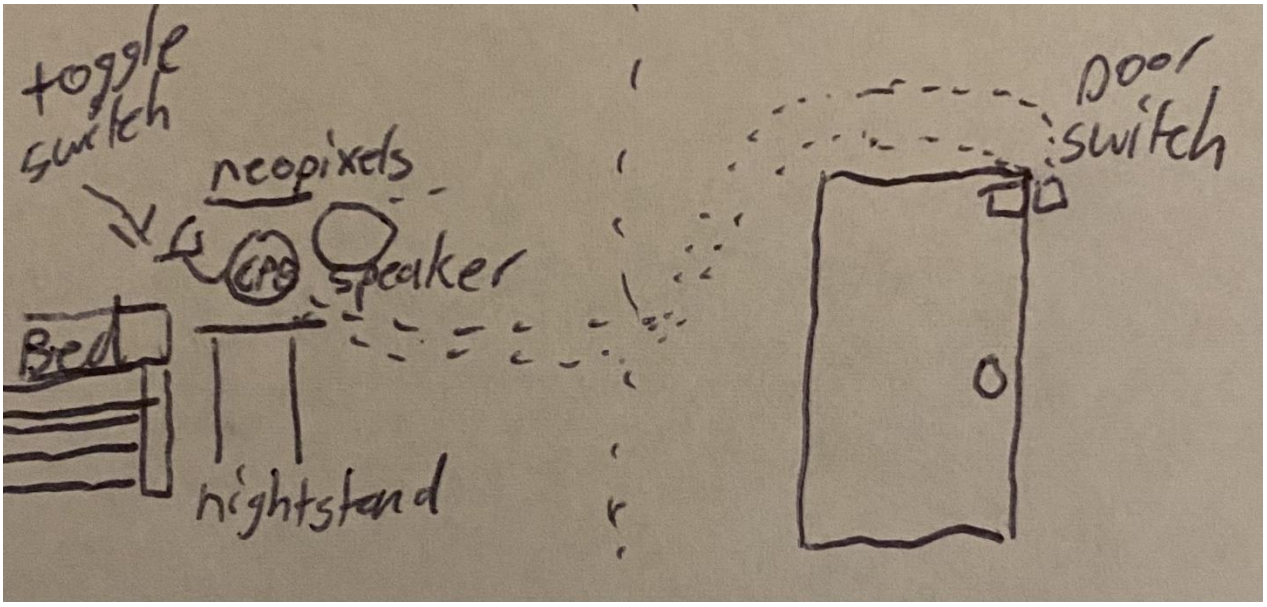
Alarm Door Detection System for a House



Presented to the Board of Instrumentation and Experimental Methods for Spring 2022: Representative Dr. Carlos Montalvo

Prepared by:

Justin Dyer

Mechanical Engineering Student, University of South Alabama

Garrison Woods

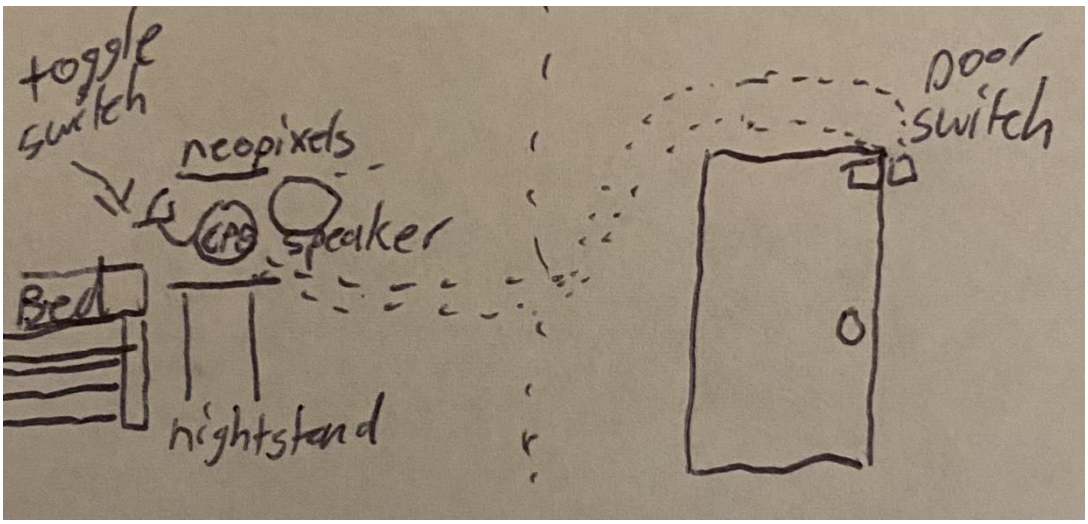
Mechanical Engineering Student, University of South Alabama

1. **Introduction**
    a. **Problem Statement**

I currently do not have a sufficient alarm system in my home. The one I do have is limited to two functions, disarmed and armed. While armed, the system will call the cops and set off an alarm sound until reset. While disarmed, the system will not do anything, basically powered off. I would prefer not to spend hundreds of dollars on an alarm system; however, this is something I want.

b. Description of Solution

The solutions to my problem can be derived into two categories, buy or make. I am restricted to a budget and buying a system would put me outside of the budget; therefore, I have decided to derive a way to satisfy an assignment I have in a class and solve a personal problem with the help of my partner. For the class I need to use 3 components along with my microcontroller. I will use a magnetic door switch to read whether the door is open or shut. I will also use a switch to toggle settings for the system and utilize LEDs on the CPB for corresponding settings. Finally, I will use an external set of LEDs and a speaker to alert myself of when the door is opened. A sketch of the idea is represented below:



c. Requirements

The specific requirements for the project are shown in the numerated list below:

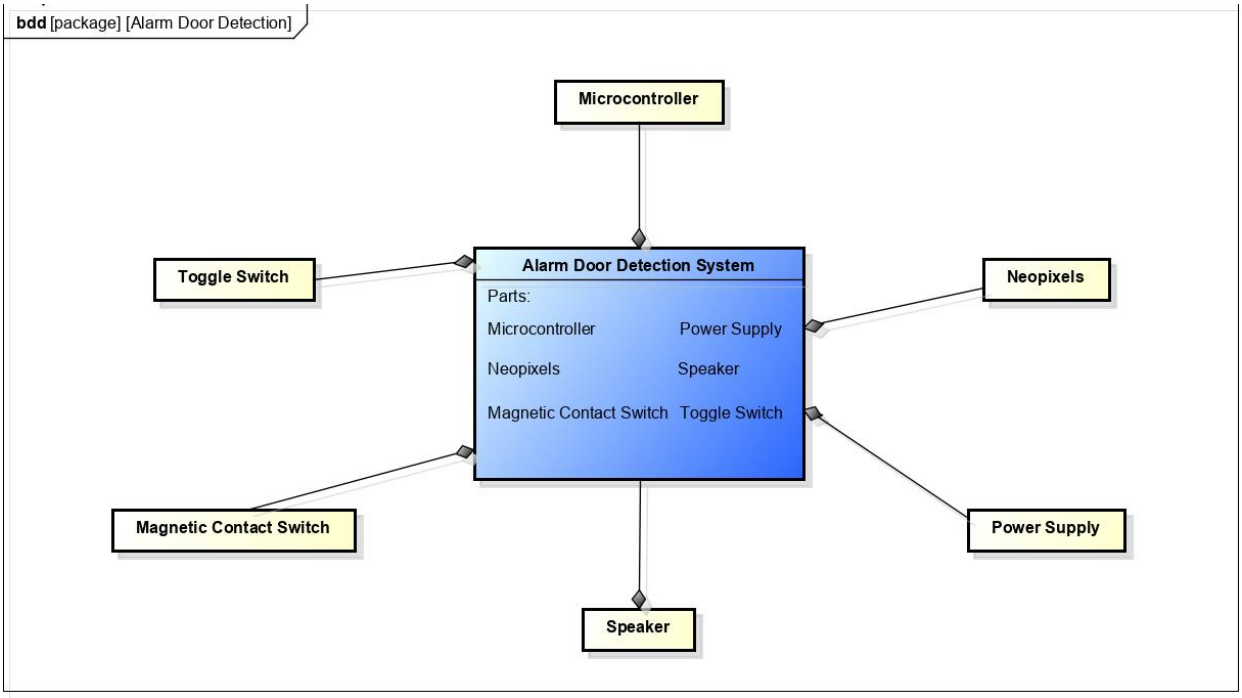
1. The system shall measure if a door is open or closed.
2. The system shall have an audio output to alert the consumer.
3. The system shall be constantly powered.
4. The system shall have LEDs to indicate settings.
5. The system shall have LEDs to alert the consumer.
6. The system shall utilize a microcontroller.
7. The system shall have 3 major components not including the battery or microcontroller.
8. The system shall have an external component not found in the instrumentation kit.
9. The system shall act as a data acquisition system.
10. The system shall be able to toggle settings.

d. Block Definition Diagram

To satisfy the requirements noted above, the following items will be needed. The numbers that follow the item represents the requirement the item satisfies.

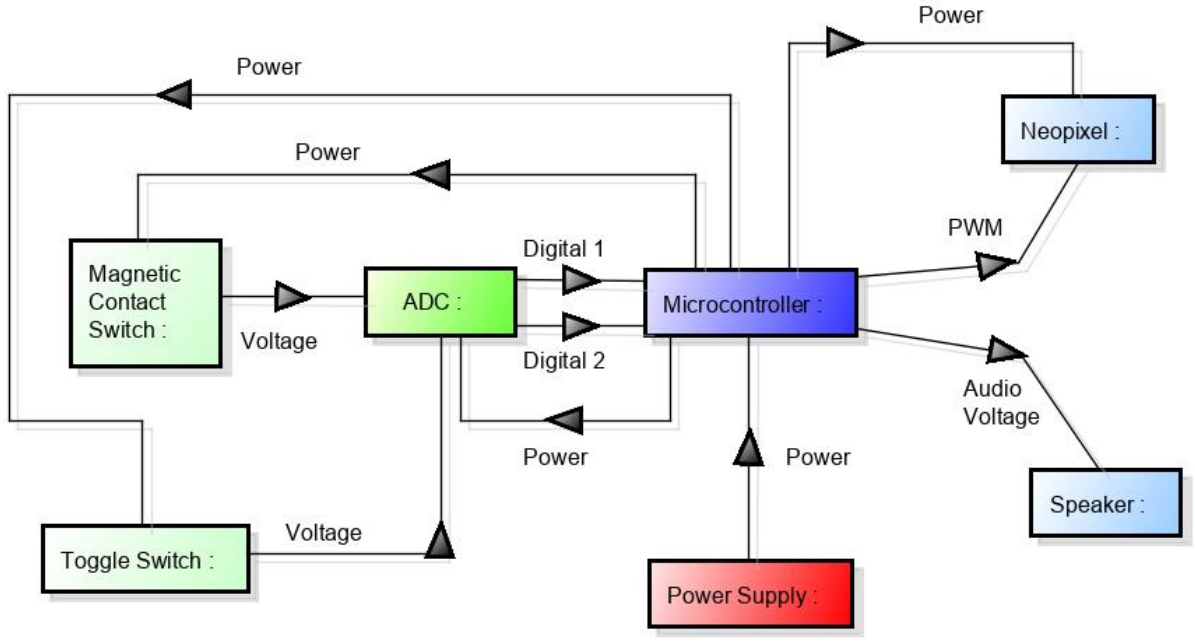
- Microcontroller: 4, 6, 9
- Power Supply: 3
- Neopixels: 5, 7, 8
- Speaker: 2, 7, 8
- Magnetic Contact Switch (door sensor): 1, 7, 8, 9
- Toggle Switch: 7, 8, 10

These components will all be running through the microcontroller, required by the course. The 4 major components are the Neopixels, Speaker, Magnetic Contact Switch, and Toggle Switch, which satisfy the requirements to have 3 major components. To show how this will be assembled, the following block definition diagram is shown below:



e. Internal Block Diagram

The diagram shown below shows how the components will be interfaced throughout the system. The power source will supply power to the microcontroller, and the microcontroller will supply power to the rest of the system. Wires will be running from the pins on the CPB to the neopixels, speaker, magnetic contact switch, and the toggle switch. The magnetic contact switch and toggle switch will send a voltage to the ADC on the CPB to convert the voltage to digital signal. The CPB will send audio signals to the speaker and PWM signals to the neopixels from the pins on the CPB.



f. Target Market

The target market for this product is very large as all home owners could make use of this product. The price for home alarm systems ranges from $250-$600 plus an average of $25 dollars per month in monthly fees. This product is budget friendly for its simplistic design and software that can be publicly released. This product would probably be sold around $75 with maybe a $25 installation fee. This price sums to $100, which is still $150 less than the lowest price in the range of competitors. The following table shows a breakdown of the system along with the components' corresponding price.

ITEM	Quantity	Price	Link
Circuit Playground Bluefruit	1	$24.95	https://www.adafruit.com/product/4333
Magnetic Contact Switch (door sensor)	1	$3.95	https://www.adafruit.com/product/375
Toggle Switch	1	$2.95	https://www.adafruit.com/product/3218
Neopixels	1	$5.95	https://www.adafruit.com/product/1426
Power Supply	1	$8.25	https://www.adafruit.com/product/1995
TOTAL		$46.05	

2. Initial Design of Prototype

a. Initial Prototype

My initial physical design of the Alarm Door Detection System has prevailed strongly thus far. For testing I assembled the same internal block diagram shown in the introduction with alligator clips.

b. First Design Change

The first design change that I faced with the Alarm Door Detection System focused on the scaling of the prototype. Initially, I was designing the prototype to be installed in my house; however, I realized that we would need to bring the prototype to class to present. I decided to scale the model down to fit in a portable box for demonstration.

Above is an image of the Alarm Door Detection System to represent the new scale I was building on.

c. Second Design Change

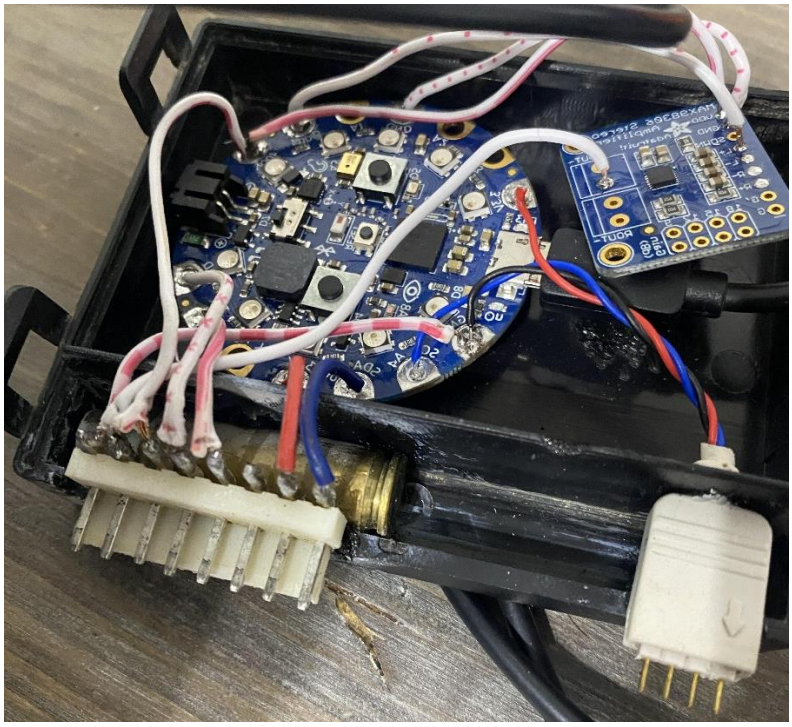
The second design change dealt more with the software side of development, rather than the hardware. Initially, when I wrote the software for the system, it would alert the user constantly when disarmed and the door was opened. I desired the disarmed setting of the system to alert the user once when the door was opened and reset when the door closes. To achieve this, I introduced a count to my loops in the software. At startup the count is zero and is changed to one when the door opens and the user is only alerted when the count is zero while disarmed.

```
74          if toggle.value is False:
75              led.value = False
76              if trip.value is False:
77                  print('Door Opened')
78                  pixels.fill((0, 100, 0))
79                  if count == 0:
80                      play_file("ALERT.wav")
81                      count = count + 1
```

Above is a snippet of the software that introduces and uses the count function.

d. Third Design Change

The last major design change was to introduce the ability to attach and detach the Bluefruit from the prototype easily. For our class, we are required to bring the Bluefruit to some classes and I needed a way to easily disassemble and reassemble my prototype. To introduce this feature, I recycled some pin attachments from old circuits. I soldered the male ends to the Bluefruit ports and the respective wires to the female ends.



Above is an image of the pin attachments soldered to the Bluefruit.

e. Thoughts for Final Design

The design I have reached thus far is fully functional and satisfies my desires for this project. It alerts the user in the way specified by the user and has yet to have a serious bug. The only changes I may make would be from peer review suggestions. One of my goals for the project is for it to be desirable. Being desirable by me does not satisfy the goal. If the product is desirable by peer reviewees, then the goal will be satisfied.

3. Presentation of Final Design

Done with the major design, I assembled a model to begin testing. I used an old casing and box I had and built a scaled door. After wiring the assembly, I would begin testing.

Above you can see my model of the Alarm Door Detection System. To test, I would set a timer for five minutes and open the door in random

intervals and record the time I opened the door. After the allotted time, I would unplug the cpb and plug it into my computer and compare data.

```
Open    Time-stamp: 3/29/2022  7:39PM
Open    Time-stamp: 3/29/2022  7:39PM
Open    Time-stamp: 3/29/2022  7:39PM
Open    Time-stamp: 3/29/2022  7:39PM
Open    Time-stamp: 3/29/2022  7:39PM
Open    Time-stamp: 3/29/2022  7:39PM
Open    Time-stamp: 3/29/2022  7:39PM
Open    Time-stamp: 3/29/2022  7:40PM
Open    Time-stamp: 3/29/2022  7:40PM
Open    Time-stamp: 3/29/2022  7:40PM
Open    Time-stamp: 3/29/2022  7:40PM
Open    Time-stamp: 3/29/2022  7:40PM
Open    Time-stamp: 3/29/2022  7:40PM
Open    Time-stamp: 3/29/2022  7:40PM
Open    Time-stamp: 3/29/2022  7:40PM
Open    Time-stamp: 3/29/2022  7:40PM
Open    Time-stamp: 3/29/2022  7:40PM
```

The figure above is a snippet of the data the cpb records. I was satisfied with the interface of the system. The only problem was with software development on the timestamp. Fighting lines of code for what felt like ages, I improved my accuracy of the timestamp to be off by 1.42μs/s. I was happy with this accuracy for the purpose of use. To improve this I would write a function that computes the full time elapsed and converts it to the timestamp, which, in turn, increases the processing speed to reduce error.

Note(must import ALERT.wav and purge.wav)

Below is my software:

Boot:

```
import board
import digitalio
import storage
```

```
# For Gemma M0, Trinket M0, Metro M0/M4 Express, ItsyBitsy M0/M4 Express
#switch = digitalio.DigitalInOut(board.D2)
# switch = digitalio.DigitalInOut(board.D5)  # For Feather M0/M4 Express
switch = digitalio.DigitalInOut(board.D7)  # For Circuit Playground Express
switch.direction = digitalio.Direction.INPUT
switch.pull = digitalio.Pull.UP

# If the switch pin is connected to ground CircuitPython can write to the drive
storage.remount("/", switch.value)
if (switch.value):
    print('Storage changed')
else:
    print('Change the switch')
```

Timestamp module:

```
import time

def Timestamp(x,y,z):
    starttime = [3,29,2022,7,39,'PM']
    minutes = starttime[4]
    hour = starttime[3]
    year = starttime[2]
    day = starttime[1]
    month = starttime[0]
    period = starttime[5]
    January = 31
    March = 31
    April = 30
    May = 31
    June = 30
    July = 31
    August = 31
    September = 30
    October = 31
    November = 30
    December = 31
    if year%400 == 0 or year%4 == 0 and year%100 !=0:
        February = 29
    else:
        February = 28
    timestamp = [month,day,year,hour,minutes,period]
    if z == 1:
        runtime = (y - x)
```

```
    if runtime > 60:
        minutes = minutes + 1
        x = time.monotonic()
    if hour < 12:
        x = 1
    if minutes > 59:
        minutes = 0
        hour = hour + 1
        if hour == 12 and x == 1:
            x = 0
            if period == 'AM':
                period = 'PM'
                pass
            if period == 'PM':
                period = 'AM'
            if hour == 12 and period == 'AM':
                day = day + 1
        if hour == 13:
            hour = 1
    if month == 1:
        length = January
    if month == 2:
        length = February
    if month == 3:
        length = March
    if month == 4:
        length = April
    if month == 5:
        length = May
    if month == 6:
        length = June
    if month == 7:
        length = July
    if month == 8:
        length = August
    if month == 9:
        length = September
    if month == 10:
        length = October
    if month == 11:
        length = November
    if month == 12:
        length = December
    if day > length:
        day = 1
```

```
        month = month + 1
    if month == 13:
        month = 1
        year = year + 1
    return str('Time-stamp: ' + str(month) + '/' + str(day) + '/' + str(year) +
'  ' + str(hour) + ':' + str(minutes) + str(period))
```

Main:

```
"""
Justin Dyer
Instrumentation project: House Alarm

"""

#Import modules
import board
import time
import digitalio
import analogio
import neopixel
from audiopwmio import PWMAudioOut as AudioOut
from audiocore import WaveFile
import Timestamp



#enabling the speaker on the board
speaker_enable = digitalio.DigitalInOut(board.SPEAKER_ENABLE)
speaker_enable.direction = digitalio.Direction.OUTPUT
speaker_enable.value = True

#Setting toggle switch as an output to toggle armed and disarmed
toggle = digitalio.DigitalInOut(board.A4)
toggle.direction = digitalio.Direction.INPUT
toggle.pull = digitalio.Pull.DOWN

#Setting board slide switch as an input to toggle data logging system
switch = digitalio.DigitalInOut(board.SLIDE_SWITCH)
switch.direction = digitalio.Direction.INPUT
switch.pull = digitalio.Pull.UP

#Setting the magnetic door switch as trip as an input
```

```
trip = digitalio.DigitalInOut(board.A5)
trip.direction = digitalio.Direction.INPUT
trip.pull = digitalio.Pull.DOWN

#Setting the D13 on the board as an output to indicate settings (may remove)
led = digitalio.DigitalInOut(board.D13)
led.direction = digitalio.Direction.OUTPUT
led.value = False

#Setting neopixels on the board as an output to indicate settings
pixels = neopixel.NeoPixel(board.TX, 8)
pixels2 = neopixel.NeoPixel(board.NEOPIXEL, 10)

#Creating a function to output audio wav file
def play_file(filename):
    print("Playing file: " + filename)
    wave_file = open(filename, "rb")
    with WaveFile(wave_file) as wave:
        with AudioOut(board.AUDIO) as audio:
            audio.play(wave)
            while audio.playing:
                pass

#making a count of the loop
count = 0

#boot up time
btime = time.monotonic()

#if in data logging mode, create TimeStamp file
if switch.value == False:
    file = open("\TimeStamp.txt","w")
else:
    print("Not opening file")

#Initial set time for time function to run off of
starttime = [3,29,2022,7,39,'PM']

#take data from start time
minutes = starttime[4]
hour = starttime[3]
year = starttime[2]
day = starttime[1]
month = starttime[0]
period = starttime[5]
```

```
#defining days in the month of interest
January = 31
March = 31
April = 30
May = 31
June = 30
July = 31
August = 31
September = 30
October = 31
November = 30
December = 31

##Try function of timestamp to develop stamp in loop


#Loop for program
while True:
    if toggle.value is True:
        ##Armed loop
        led.value = True
        if trip.value is False:
            doorvalue = 'Open'
            while True:
                print('Intruder')
                pixels2.fill((100, 0, 0))
                pixels.fill((100, 0, 0))
                play_file("purge.wav")
                if toggle.value is False:
                    break
        if trip.value is True:
            doorvalue = 'Closed'
            print('Safe')
            pixels2.fill((0, 0, 0))
            pixels.fill((0, 0, 0))
    if toggle.value is False:
        ##Disarmed loop
        led.value = False
        if trip.value is False:
            doorvalue = 'Open'
            print('Door Opened')
            pixels2.fill((0, 100, 0))
            pixels.fill((0, 100, 0))
            if count == 0:
```

```
                play_file("ALERT.wav")
                count = count + 1
        if trip.value is True:
            doorvalue = 'Closed'
            print('Door Closed')
            pixels2.fill((0, 0, 0))
            pixels.fill((0, 0, 0))
            if count == 1:
                count = count - 1
    pixels.show()
    ##Check if it is a leap year and assign days of Feb.
    if year%400 == 0 or year%4 == 0 and year%100 !=0:
        February = 29
    else:
        February = 28
    timestamp = [month,day,year,hour,minutes,period]
    #The time in seconds since boot
    runtime = (time.monotonic() - btime)

    ##Calendar addition
    if runtime > 60:
        minutes = minutes + 1
        btime = time.monotonic()
    if hour < 12:
        x = 1
    if minutes > 59:
        minutes = 0
        hour = hour + 1
        if hour == 12 and x == 1:
            x = 0
            if period == 'AM':
                period = 'PM'
                pass
            if period == 'PM':
                period = 'AM'
            if hour == 12 and period == 'AM':
                day = day + 1
        if hour == 13:
            hour = 1
    if month == 1:
        length = January
    if month == 2:
        length = February
    if month == 3:
        length = March
```

```
    if month == 4:
        length = April
    if month == 5:
        length = May
    if month == 6:
        length = June
    if month == 7:
        length = July
    if month == 8:
        length = August
    if month == 9:
        length = September
    if month == 10:
        length = October
    if month == 11:
        length = November
    if month == 12:
        length = December
    if day > length:
        day = 1
        month = month + 1
    if month == 13:
        month = 1
        year = year + 1

    ##if in data logging setting, write in the file
    ##Must have switch = false upon boot
    if switch.value == False:
        output = str(str(doorvalue) + '     ' + str('Time-stamp: ' + str(month) +
'/' + str(day) + '/' + str(year) + '   ' + str(hour) + ':' + str(minutes) +
str(period))  + str('\n'))
        file.write(output)
        file.flush()
    if switch.value == True:
        output = "Not writing data to file"
    print(output)
    #print((int(trip.value), ))
    #print(count)
    #print(time.monotonic())
    time.sleep(0.1)
```