# Ultrasonic Sensor (MakeCode Python)

**“””**

**1. display and send detected distance value to raspberry pi.**

**2. set sound variable (To determine if alarm needs to play).**

**“””**

sound = 0

display = ""

# create queue of values to send over radio.

# Set consistent radio group between all sensors

sendQueue: List[str] = []

radio.set\_group(1)

# display recorded values on screen

def on\_forever():

global display

while len(sendQueue) > 0:

display = sendQueue.shift()

basic.show\_string("" + (display))

radio.send\_string("" + (display))

basic.forever(on\_forever)

# Set sound variable to detect if distance from hoard is acceptable

# if detected distance is below 10cm; hoard is too near to sensor

def on\_forever2():

global sound

if grove.measure\_in\_centimeters(DigitalPin.P0) > 10:

sound = 0

else:

sound = 1

basic.forever(on\_forever2)

**“””**

**If detected distance is not acceptable:**

**Activates sound and red bulb proportional to distance detected from hoard. Additional ‘elseif’ for when hoard touches sensor.**

**Turns OFF green bulb and sends value to queue with string ‘UTL’.**

**Waits a set amount of time between each measurement. (5s here)**

**If detected distance is acceptable:**

**Turns OFF red light and turns ON green light.**

**“””**

def on\_forever3():

if sound == 1 and grove.measure\_in\_centimeters(DigitalPin.P0) > 0:

sendQueue.append("UTL:" + convert\_to\_text(grove.measure\_in\_centimeters(DigitalPin.P0)))

pins.digital\_write\_pin(DigitalPin.P2, 0)

pins.analog\_write\_pin(AnalogPin.P1,

1 / Math.ceil(grove.measure\_in\_centimeters(DigitalPin.P0)) \* 1000)

music.play\_tone(1 / Math.ceil(grove.measure\_in\_centimeters(DigitalPin.P0)) \* 1000,

music.beat(BeatFraction.WHOLE))

basic.pause(5000)

elif sound == 1 and grove.measure\_in\_centimeters(DigitalPin.P0) == 0:

sendQueue.append("UTL:" + convert\_to\_text(grove.measure\_in\_centimeters(DigitalPin.P0)))

pins.digital\_write\_pin(DigitalPin.P2, 0)

pins.analog\_write\_pin(AnalogPin.P1, 1023)

music.play\_tone(1500, music.beat(BeatFraction.WHOLE))

basic.pause(5000)

else:

pins.digital\_write\_pin(DigitalPin.P1, 0)

pins.digital\_write\_pin(DigitalPin.P2, 1)

basic.forever(on\_forever3)

# 

# Weight Sensor (MakeCode Python)

**“””**

**Send detected weight over radio with string ‘WGT’ for identification. If weight exceeds limit, play alarm sound.**

**Waits a set amount of time between each measurement. (5s here)**

**“””**

# Initialise weight sensor to pins P1 and P2

weight = 0

SGBotic.init\_loadcell(DigitalPin.P1, DigitalPin.P2)

SGBotic.loadCell\_noLoad()

# Set consistent radio group between all sensors

radio.set\_group(1)

def on\_forever():

global weight

weight = SGBotic.read\_grams()

basic.show\_number(weight)

radio.send\_string("WGT:" + convert\_to\_text(weight))

basic.pause(5000)

basic.forever(on\_forever)

# Clear Screen when Button A is pressed

def on\_button\_pressed\_a():

basic.clear\_screen()

input.on\_button\_pressed(Button.A, on\_button\_pressed\_a)

# Calibrate Sensor to 0 grams when Button B is pressed

def on\_button\_pressed\_b():

SGBotic.cali\_UserLoad(0)

basic.show\_string("D")

input.on\_button\_pressed(Button.B, on\_button\_pressed\_b)

# Bluetooth Dongle (MakeCode Javascript)

enum RadioMessage {

message1 = 49434

}

# Serial Port Settings and Radio Settings

# Set consistent radio group between all sensors

serial.redirect(

SerialPin.USB\_TX,

SerialPin.USB\_RX,

BaudRate.BaudRate115200

)

radio.setGroup(1)

# Check if values received are from sensors

radio.onReceivedString(function (receivedString) {

serial.writeLine(receivedString)

if (receivedString.includes("UTL")) {

basic.showLeds(`

# . . . #

# . . . #

# . . . #

# . . . #

. # # # .

`)

}

if (receivedString.includes("WGT")) {

basic.showLeds(`

# . # . #

# . # . #

# . # . #

# . # . #

. # . # .

`)

}

})

# Raspberry Pi to Database (Postgresql database) (Python)

import os

import time

# Postgresql module

import psycopg2

from datetime import datetime

from serial import Serial

# Hosted Database URL

DATABASE\_URL = '\_\_\_\_\_\_\_\_\_\_\_\_\_'

**“””**

**Function to read incoming messages from bluetooth dongle microbit, received through serial. Message validation is in the form of ‘UTL:’ or ‘WGT:’ to ensure they are from the sensors. If messages are in the correct form, connect to the database and insert the data.**

**“””**

def msgRead(receivedMsg):

if ( (len(receivedMsg) >= 4) and (receivedMsg[3] == b':'[0])):

conn = psycopg2.connect(DATABASE\_URL, sslmode='require')

cur = conn.cursor()

msgType = receivedMsg[0:3]

msgData = receivedMsg[4:]

# Message from Ultrasonic Sensor Microbit, along with time and date

if ( msgType == b'UTL' ):

timeString = datetime.now().strftime('%H:%M:%S')

clock = 'TIM:' + timeString

dateString = datetime.now().strftime('%d-%b-%Y')

date = 'DAT:' + dateString

ultrasonic = 'UTL:' + msgData.decode('ascii')

print(clock)

print(date)

print(ultrasonic)

“””

Converts received data in bytes and converts it into string.

Then inserts data into database ultrasonic table

“””

cur.execute('''insert into ultrasonic (date,time,distance)

values (%s,%s,%s) ; ''',

(dateString,timeString,float(msgData.decode('ascii'))))

# Message from Weight Sensor Microbit, along with time and date

elif ( msgType == b'WGT' ):

timeString = datetime.now().strftime('%H:%M:%S')

clock = 'TIM:' + timeString

dateString = datetime.now().strftime('%d-%b-%Y')

date = 'DAT:' + dateString

weight = 'WGT:' + msgData.decode('ascii')

print(clock)

print(date)

print(weight)

“””

Converts received data in bytes and converts it into string.

Then inserts data into database weight table

“””

cur.execute('''insert into weight (date,time,weight)

values (%s,%s,%s) ; ''',

(dateString,timeString,float(msgData.decode('ascii'))))

else:

print("Unknown Data")

# Commit changes into database then close database connection

conn.commit()

cur.close()

conn.close()

“””

Code taken from <https://micromag.cc/serial-communications-with-micro-bit-raspberry-pi/> and simplified for our needs. Code used to read the serial ports on raspberry pi and then connect to microbit and receive data.

“””

nextCompassPoll = 0.0 ;

serialDevDir='/dev/serial/by-id'

if ( os.path.isdir(serialDevDir) ):

serialDevices = os.listdir(serialDevDir)

if ( len(serialDevices) > 0 ):

serialDevicePath1 = os.path.join(serialDevDir, serialDevices[0])

#serialDevicePath2 = os.path.join(serialDevDir, serialDevices[1])

serial1 = Serial(port=serialDevicePath1, baudrate=115200, timeout=0.2)

#serial2 = Serial(port=serialDevicePath2, baudrate=115200, timeout=0.2)

while( True ):

receivedMsg1 = serial1.readline()

#receivedMsg2 = serial2.readline()

msgRead(receivedMsg1)

#msgRead(receivedMsg2)

currentTime = time.time()

if ( currentTime > nextCompassPoll ):

serial1.write(b'CMP:\n')

#serial2.write(b'CMP:\n')

nextCompassPoll = currentTime + 2.0

else:

print('No serial devices connected')

else:

print(serialDevDir + ' does not exist')

# Telegram Bot (Python)

**“””**

**Code for running the telegram bot through python taken from** [**https://pythoncircle.com/post/265/how-to-create-completely-automated-telegram-channel-with-python/**](https://pythoncircle.com/post/265/how-to-create-completely-automated-telegram-channel-with-python/)

**Telegram bot created using botfather on telegram**

**“””**

import telegram

import psycopg2

import time

import schedule

import \_thread

# function to run the telebot

def timeDelay():

truth = False

# use token generated when creating telegram bot via botfather.

bot = telegram.Bot(token="\_\_\_\_\_\_\_\_")

**“””**

**Connect to database and select the last 8 data entries.**

**Print out all measured values in the 8 rows and set variable to True if values exceed set limit i.e. when alarm needs to be raised.**

**Link same database URL as previous.  
“””**

DATABASE\_URL ='\_\_\_\_\_\_\_\_\_\_\_\_\_'

conn = psycopg2.connect(DATABASE\_URL, sslmode='require')

sql ='''SELECT \* FROM ultrasonic ORDER BY date,time DESC LIMIT 8'''

cur= conn.cursor()

cur.execute(sql)

rows = cur.fetchall()

for row in rows:

num = row[2]

print(num)

if num < 5:

truth = True

**“””**

**If exceed, send message to telegram channel via bot. Chat\_id = name of telegram channel. Close database connection after.**

**“””**

if truth == True:

status = bot.send\_message(chat\_id="@CS460AlertTest", text="Time to tidy! Area is too messy!! Please send help.", parse\_mode=telegram.ParseMode.HTML)

#Closing the current connection

cur.close()

#Closing the connection

conn.close()

# Setup timer every(\_\_\_ seconds) to repeat above function

schedule.every(30).seconds.do(timeDelay)

while True:

# Checks whether a scheduled task

# is pending to run or not

schedule.run\_pending()

time.sleep(1)