# Moodmusic

Anthony Lam and Sally Kim

# Design Concept

A music player that automatically chooses a music to play for the user depending on the user's mood, defined by biometrics and environment.

# Key Parts

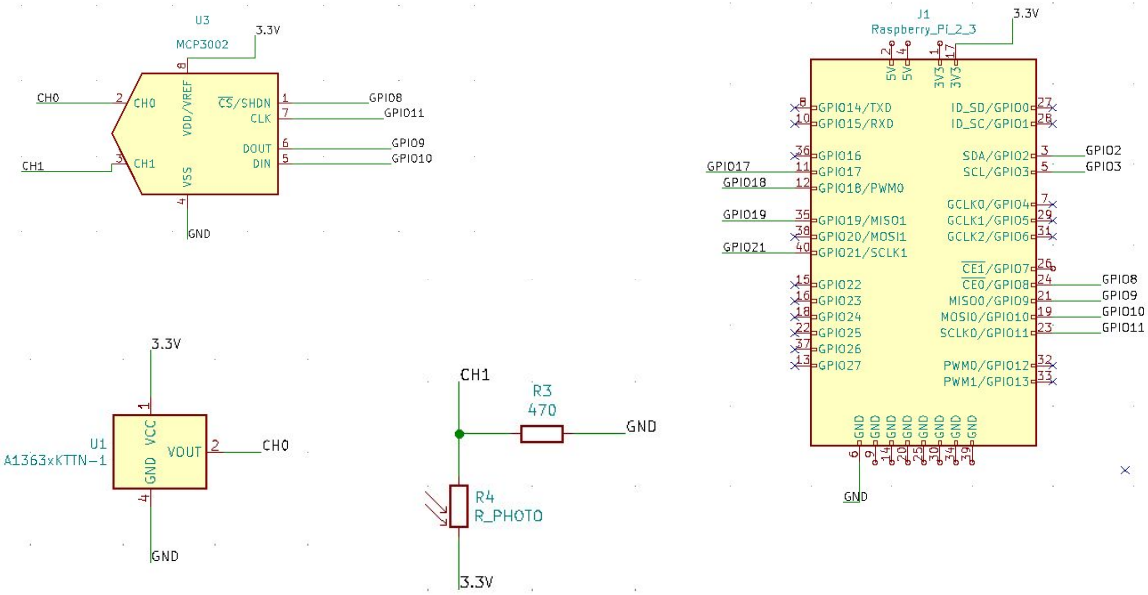
Sensors:

- Temperature (DS18B20)
- Motion (ADXL345)
- Light (Photoresistor)
- Heart Rate

Others:

- ADC (MCP3002)
- Audio Jack (Stereo Decoder)

# Circuits - Analog Sensors

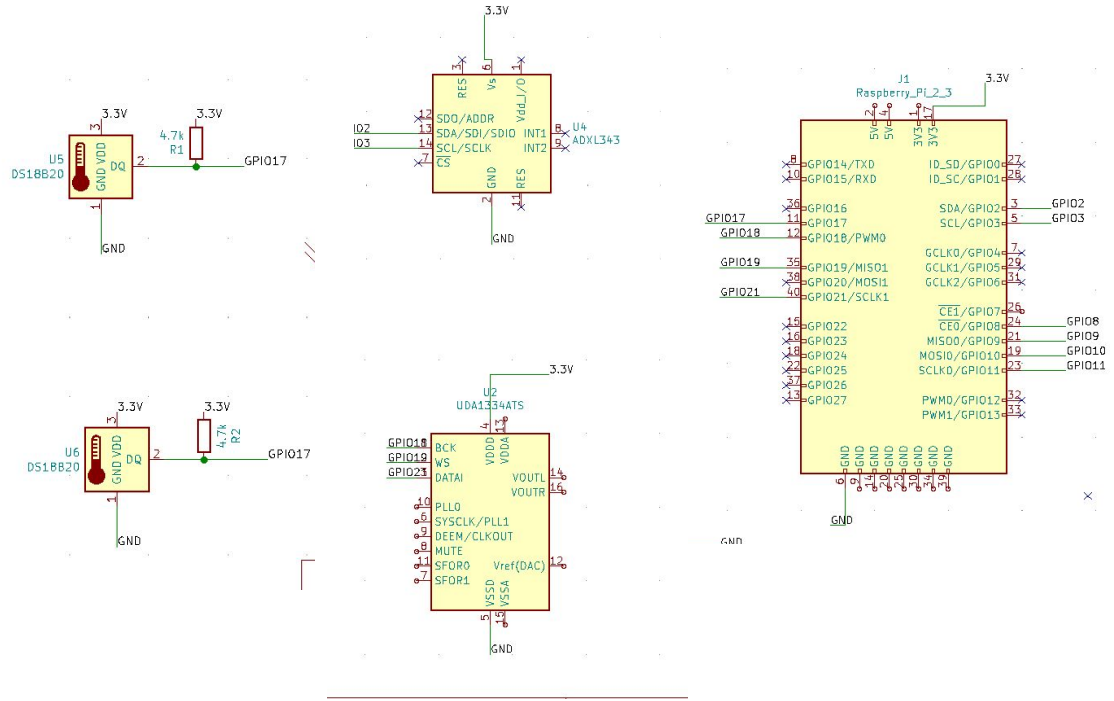



Ch0 - Heart rate sensor

Ch1 - Photoresistor

# Circuits - Digital Sensors

Temperature: One-wire interface (daisy chained)



# Library Functions

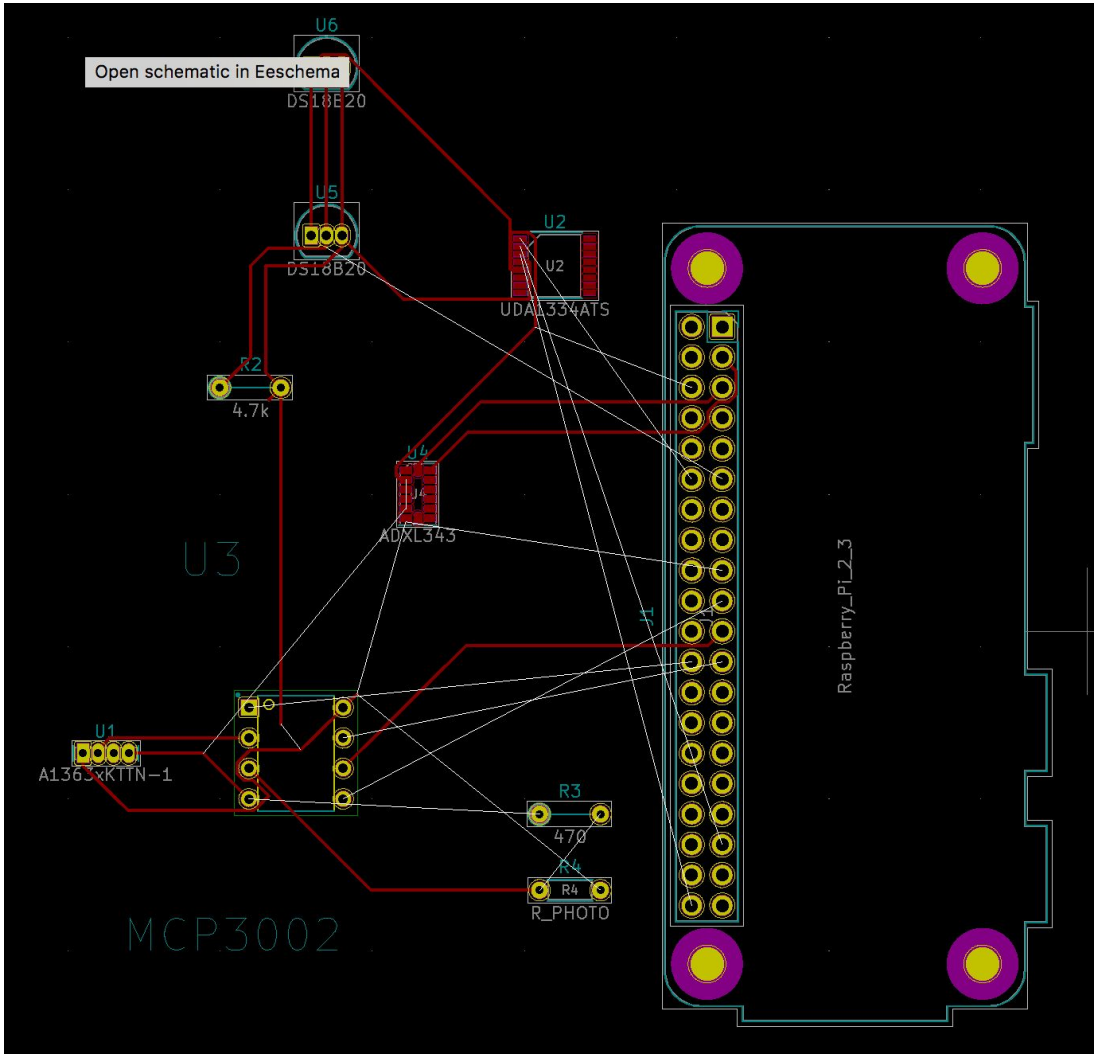
```
class Pulsesensor:
    def __init__(self, channel = 0, bus = 0, port = 0):
        self.channel = channel
        self.BPM = 0
        self.adc = MCP3002(bus, port)

    def getBPMLoop(self):
        # init variables
        rate = [0] * 10          # array to hold last 10 IBI
        sampleCounter = 0        # used to determine pulse t
        lastBeatTime = 0         # used to find IBI
        P = 512                  # used to find peak in puls
        T = 512                  # used to find trough in pu
        thresh = 525             # used to find instant mome
        amp = 100                # used to hold amplitude of
        firstBeat = True         # used to seed rate array s
        secondBeat = False       # used to seed rate array s

        IBI = 600                # int that holds the time i
        Pulse = False            # "True" when User's live h
        lastTime = int(time.time()*1000)
```

```
def accelerometer():
    i2c = busio.I2C(board.SCL, board.SDA)
    accelerometer = adafruit_adxl34x.ADXL345(i2c)
    return accelerometer
    #code and setup from https://pimylifeup.com/rasp
```

```
class MCP3002:
    def __init__(self, bus=0, port= 0):
        self.bus = bus
        self.port = port
        self.open()

    def open(self):
        self.spi = spidev.SpiDev()
        self.spi.open(self.bus,self.port)
        self.spi.max_speed_hz = 7629
    #photoresistor
    def read(self, adc_channel):
        resp = self.spi.xfer2([0b01100000 + (adc_channel << 4), 0b00000000])

        counter = 0;
        value = 0
        for b in resp:
            if counter == 0:
                value += (b<<8)
                counter+=1
            else: value+=b
        return value
```

```
def temperature_sensor():
    # Initialize the GPIO Pins
    os.system('modprobe w1-gpio')  # Turns on the GPIO module
    os.system('modprobe w1-therm') # Turns on the Temperature module

    # Finds the correct device file that holds the temperature data
    base_dir = '/sys/bus/w1/devices/'
    device_folder = glob.glob(base_dir + '28*')[0]
    device_folder2 = glob.glob(base_dir+'28*')[1]
    device_file = device_folder + '/w1_slave'
    device_file2 = device_folder2 + '/w1_slave'
    value1 = read_temp(device_file)
    value2 = read_temp(device_file2)
    return value1, value2
```

# PCB layout


Open schematic in Eeschema
U6
DS18B20
U5
DS18B20
U2
U2
UDA1334ATS
R2
4.7k
U4
U4
ADXL343
U3
MCP3002
U1
A1363xKTTN-1
R3
470
R4
R4
R_PHOTO
J1
Raspberry_Pi_2_3