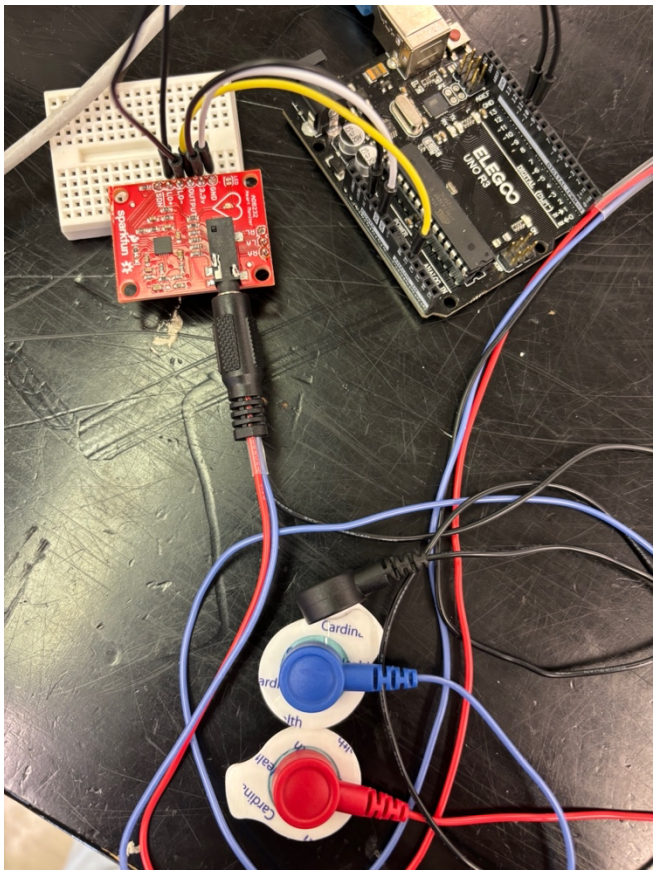
Building an ECG Machine
Microcontrollers Final Lab Notebook
Jax Lubkowitz

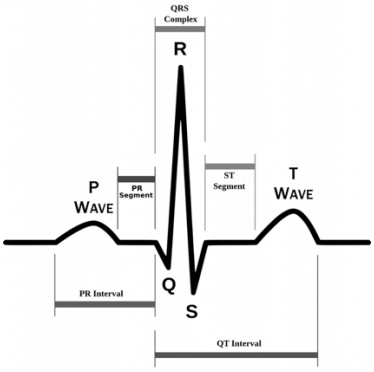
April 16, 2024
Original Inspiration - https://www.instructables.com/ECG-Monitoring-System-by-Using-Arduino-or-AD8232/

ECG machines attach to chest and measure heart rate variations.

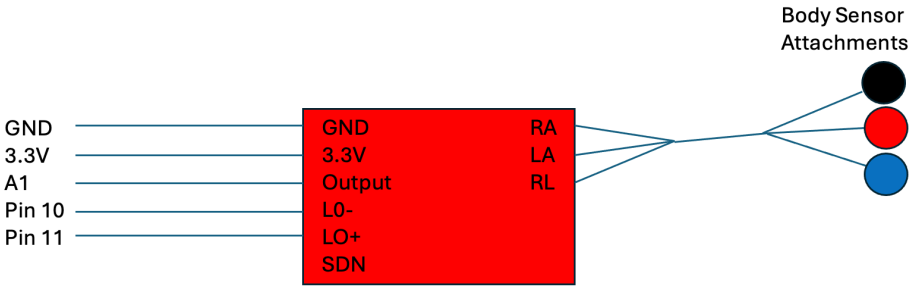
Tools needed –
Arduino
Disposable nodes
Wire with 3 node attachments to 1 male connecter to connect nodes to machine.

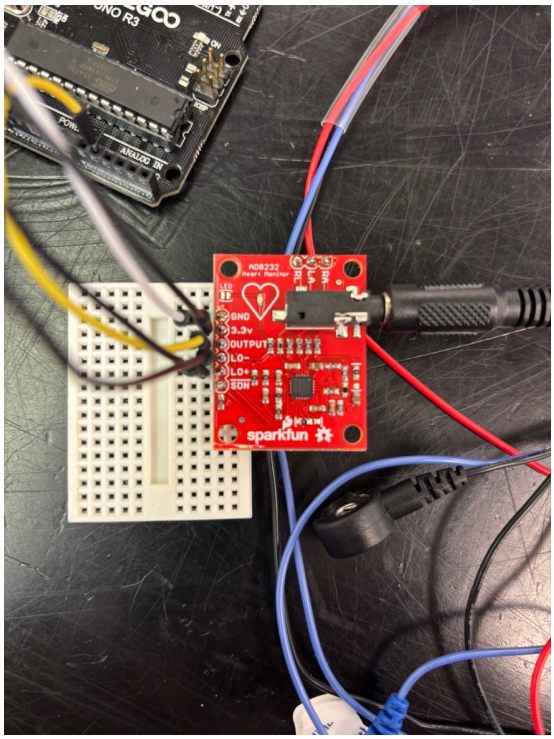


Project goals and Objective –
Build a small lightweight ECG machine that can take accurate readings. Different shapes can indicate different underlaying problems.



Assembling the board –

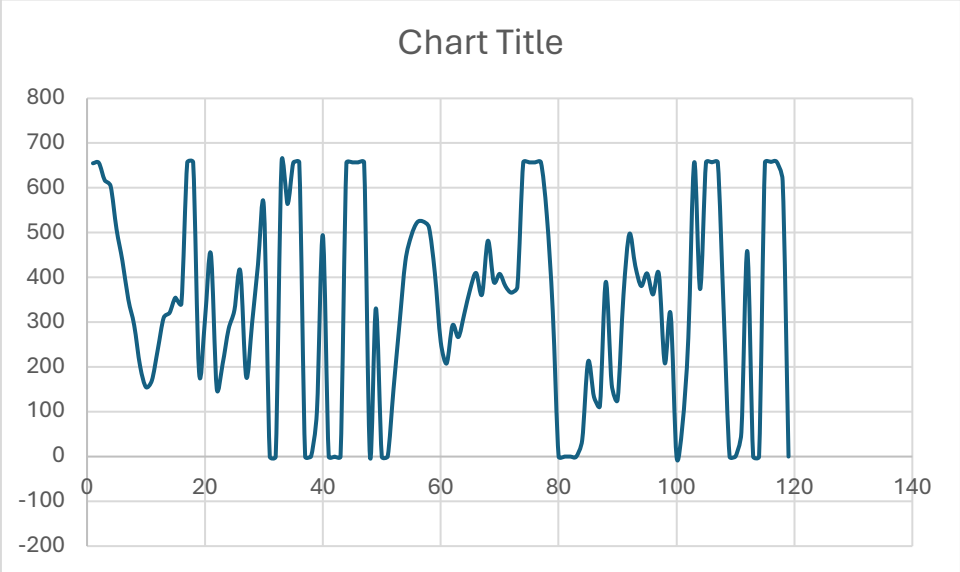




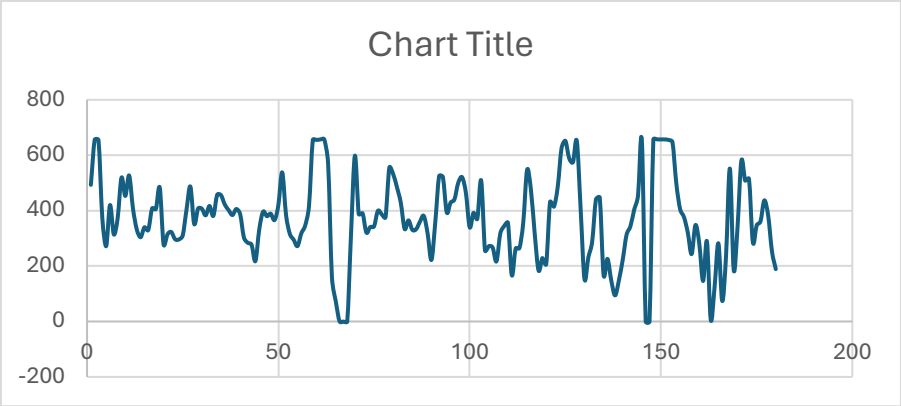
Run wires from ECG Module (AD8232) to Arduino (10 for L0+ and 11 for L0-) rest self-explanatory, Output to A1
Now plugged in controllers and used this code

```
void setup() {
  pinMode(10,INPUT);
  pinMode(11,INPUT);
pinMode(A1, INPUT);

}

void loop() {
  Serial.begin(9600);
  if((digitalRead(10)==1)||(digitalRead(11)==1)){
     Serial.println("Gagal");
  }
  else{
     Serial.println(analogRead(A1));
  }
  delay(50);
  Serial.end();
  delay(50);
}
```

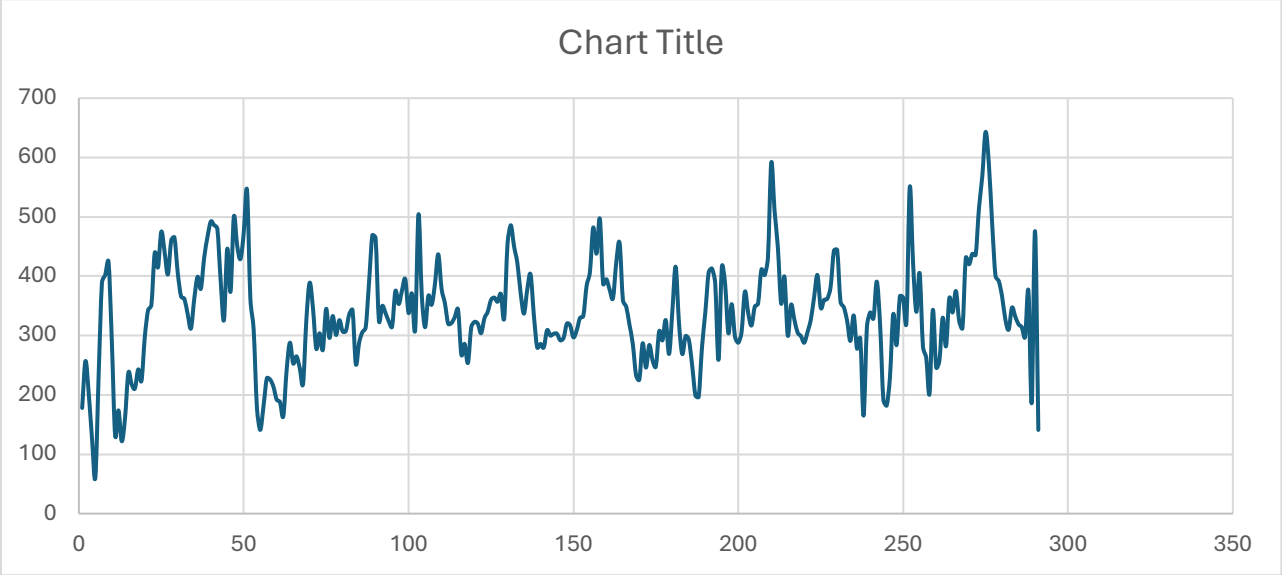
Put nodes on left chest near arm, right chest near arm and right above heart.
Ran it and signal looked like this.



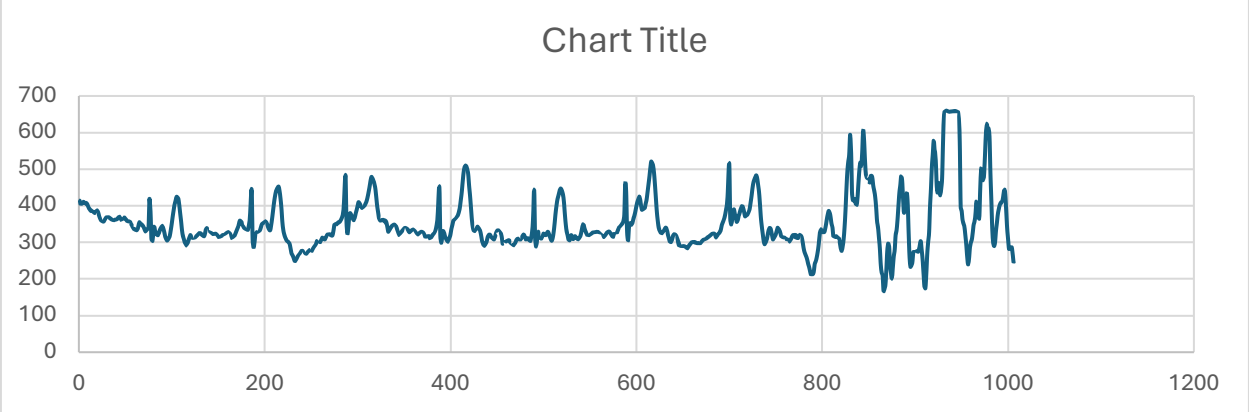
Reduced delay from 100 ms to 50 for clearer peaks and more accuracy



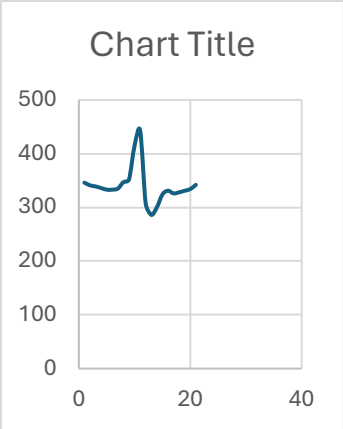
Reduced delay again to 25



Reduced Delay again to 10

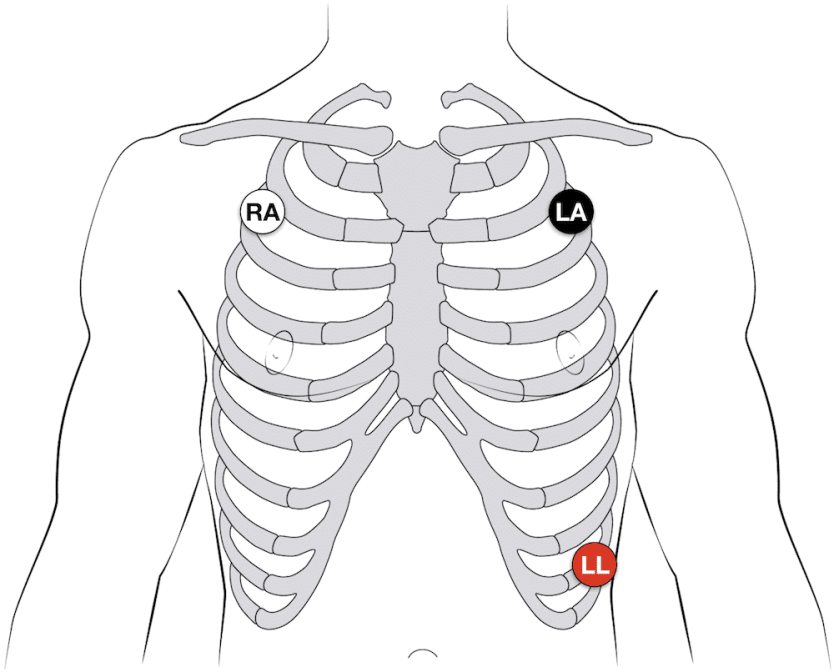


Isolating one cycle with minimal noise



April 18, 2024

Investigate best locations for 3 sensor ecg



https://litfl.com/ecg-lead-positioning/

Tried to use processing IDE (in java to build visual)
Code wasn't great switching to python

```
import processing.serial.*;

Serial myPort;       // The serial port
int xPos = 1;        // horizontal position of the graph
```

```
float height_old = 0;
float height_new = 0;
float inByte = 0;


void setup () {
  height = 400;
  size(1000, 400);   // set the window size:

  // List all the available serial ports
  println(Serial.list());

  myPort = new Serial(this, Serial.list()[1], 9600);// Open whatever port is the one you're
using.
  myPort.bufferUntil('\n');   // don't generate a serialEvent() unless you get a newline
character:
  // set inital background:
  background(0xff);
}


void draw () {
  // everything happens in the serialEvent()
}


void serialEvent (Serial myPort) {

  String inString = myPort.readStringUntil('\n');// get the ASCII string:

  if (inString != null) {

    inString = trim(inString);// trim off any whitespace:

    // If leads off detection is true notify with blue line
    if (inString.equals("!")) {
      stroke(0, 0, 0xff); //Set stroke to blue ( R, G, B)
      inByte = 512;  // middle of the ADC range (Flat Line)
    }
    // If the data is good let it through
    else {
      stroke(0xff, 0, 0); //Set stroke to red ( R, G, B)
      inByte = float(inString);
      //System.out.println(inByte);
```

```
  }
  if (Float.isNaN(inByte)) { // Check if the value is not NaN
    inByte = 512; // Assign the valid value to sensorValue
    System.out.println("Test");
  }
  //Map and draw the line for new data point
  inByte = map(inByte, 0, 1023, 0, height);
  //System.out.println(inByte);
  height_new = height - inByte;
  line(xPos - 1, height_old, xPos, height_new);
  height_old = height_new;

  // at the edge of the screen, go back to the beginning:
  if (xPos >= width) {
    xPos = 0;
    background(0xff);
  }
  else {
    // increment the horizontal position:
    xPos++;
  }

 }
}
```

Python Attempt - Still working on...

```
import serial
import matplotlib.pyplot as plt
from collections import deque

# Initialize serial connection with Arduino
ser = serial.Serial('/dev/cu.usbmodem14101', 9600)  # Replace 'COM3' with
your Arduino's serial port




# Initialize a deque to store data points
x_data = deque(maxlen=100)  # Adjust maxlen according to how many data points
you want to display
y_data = deque(maxlen=100)
# Create a plot
plt.ion()  # Turn on interactive mode
fig, ax = plt.subplots()
line, = ax.plot(x_data, y_data)
ax.set_xlabel('Time')
ax.set_ylabel('Value')

# Start reading and plotting data
```

```
try:
    while True:
        data = ser.readline().decode().strip()  # Read data from Arduino
        value = float(data)  # Convert data to float
        x_data.append(len(x_data))  # Append new x value (time)
        y_data.append(value)  # Append new y value (Arduino data)

        # Update plot
        line.set_xdata(x_data)
        line.set_ydata(y_data)
        ax.relim()
        ax.autoscale_view()

        # Redraw the plot
        plt.draw()
        plt.pause(0.001)  # Pause to allow time for the plot to update
except KeyboardInterrupt:
    print("Plotting stopped by user")

# Close serial connection
ser.close()
```

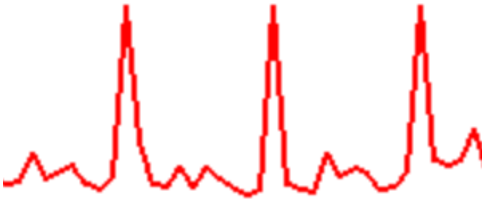
April 23rd, 2021

Playing with python window to update
Trying to get an accurate live visual, with wrapping around
After much trial and error got this code that works.
Switched to printing pixels and then drawing lines between them

```
import pygame
import sys
import serial

ser = serial.Serial('/dev/cu.usbmodem14101', 9600)  # Replace 'COM3' with
your Arduino's serial port
# Initialize pygame
pygame.init()

# Set up the window
window_width = 1000
window_height = 1000
window = pygame.display.set_mode((window_width, window_height))
window.fill((255, 255, 255))  # Fill with white color
pygame.display.set_caption("ECG")

# Define a pixel color (red)
pixel_color = (255, 0, 0)  # Red color

# Define the starting position of the pixel

pixel_x = 0
pixel_y = window_height // 2  # Center of the window vertically
pixel_x_prev = pixel_x
pixel_y_prev = pixel_y
# Define the pixel velocity (movement speed)
pixel_velocity = 5  # Pixels per frame
```

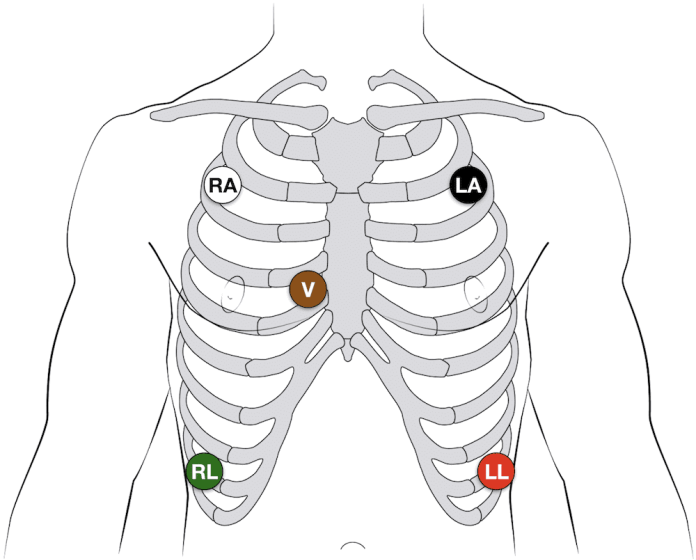
```
# Define the game loop
running = True
while running:
    # Check for events (e.g. quit)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # Move the pixel horizontally
    pixel_x += pixel_velocity

    # If the pixel goes past the right side of the window, reset its position
    if pixel_x >= window_width:
        pixel_x = 0
        pixel_x_prev = 0
        # Clear the window
        window.fill((255, 255, 255))  # Fill with white color

    data = ser.readline().decode().strip()  # Read data from Arduino
    data = 1023 - float(data)
    scale = (data) / (1023)

    # Map the value to the output range
    pixel_y = scale * (200) + 500
    # Change the color of the single pixel
    window.set_at((pixel_x, int(pixel_y)), pixel_color)

    pygame.draw.line(window, pixel_color, (pixel_x_prev, pixel_y_prev),
(pixel_x, pixel_y), 2)
    pixel_x_prev = pixel_x
    pixel_y_prev = pixel_y

    # Update the display
    pygame.display.update()

    # Control the frame rate (e.g., 9600 frames per second)
    pygame.time.Clock().tick(9600)

# Quit pygame and clean up resources
pygame.quit()
sys.exit()
```

Got this read from python window – only clean signal

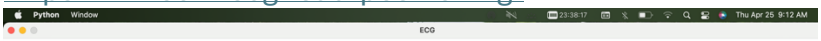


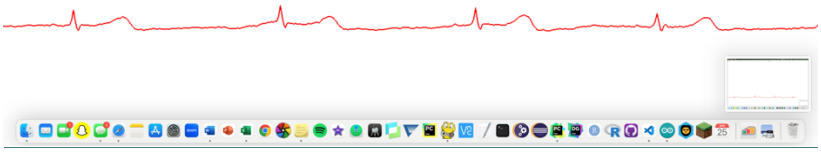
Signal is very messy and depends on user not moving and holding breath....
Next class going to try and collect really clean data using new probes and cleaning spots before application.

April 25th, 2024
Goal: try to get very accurate measurements
Used new ECG pads and cleaned attachment points with isopropyl alcohol
Used ohm meter to figure out which pad is connected to which location on board
Black = RA
Blue = LA
Red = RL

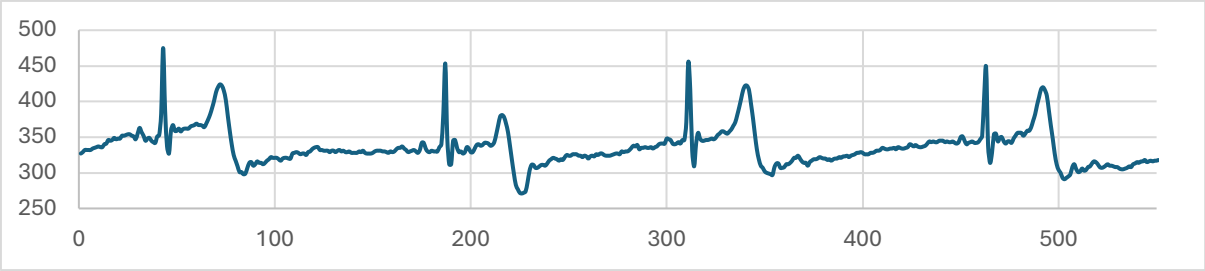


https://litfl.com/ecg-lead-positioning/





Excel Version



Had some serious success with cleaning spots
Data was very good and visual worked super well
Collected video for final presentation and also data/visuals

April 30, 2024
Working on commenting in code for ease of reading
Also playing around with scaling feature by changing the 'scale' variable in code but did not produce nice looking results. Sticking with what I had initially.
Working on final presentation – building outline and beginning to fill it in. See PowerPoint for presentation development.