Biomedical Engineering
DIY ECG

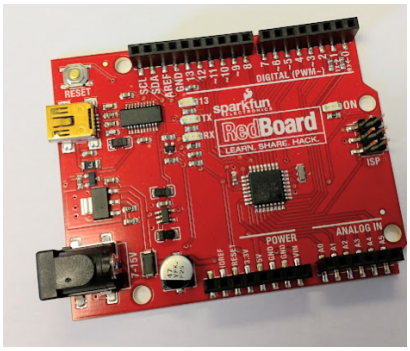
Dr. Samuel Bechara
Marquette University

Objective

This guide will walk you through the steps necessary to construct and program an electrocardiogram (ECG) machine using an ardunio, heart rate sensor module, and the arduino programming language. This single lead system will allow students to visualize the electrical activity of their heart. Students will take measurements before and after physical activity and will analyze the differences of their ECG. The goal is to gain exposure to fundamental biomedical engineering principles such as, heart physiology, electrical hardware, software design, and biometric data analysis.

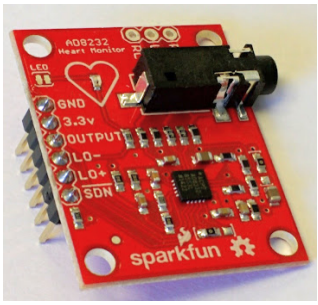
Materials

SparkFun Redboard (Arduino microcontroller)



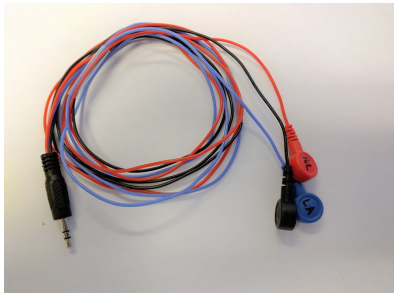
This is the “brain” of the whole project and is a very versatile and powerful computer! It can receive and/or send data in a number of different ways. In this lab we are going to concentrate on the analog output from the heart rate monitor but the capabilities of this are only limited by your imagination!

SparkFun Single Lead Heart Rate Monitor



This is what receives the electrical signal from the electrodes that are connected your body, processes the information, and then sends that as an analog signal to the Redboard. The headers that allow it to connect to the breadboard should already be soldered on for you.

Sensor Cable - Electrode Pads

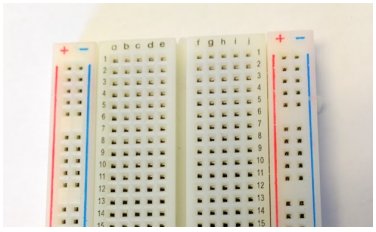


This cable connects to the heart rate monitor on one end, and to the electrodes on the other. You can think of this as a wire that runs from the electrodes to the heart rate monitor.

Electrodes

These sticky pads connect to your body and are the interface between man and machine. They will pick up the electrical activity from your heart. The sticky gel is actually conductive and allows the electrical signal to travel more easily.

Breadboard

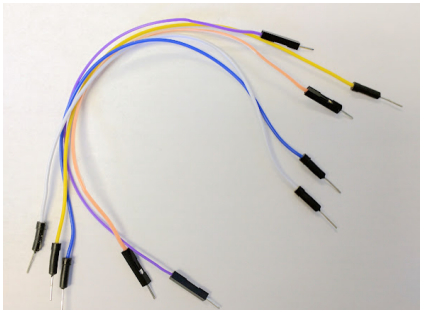


This is a very common device used as a prototyping helper. It allows us to make electrical connections easily.

USB Mini-B cable

This cable allows us to transfer our software from the computer, to the ardunio. It also allows the arduino to draw power from the computer.

Male-to-male jumper wires



These are just helpful wires to help us make electrical connections from all of the devices.

Part 1: Hardware and Electronics Assembly

Background – Basic Electronics

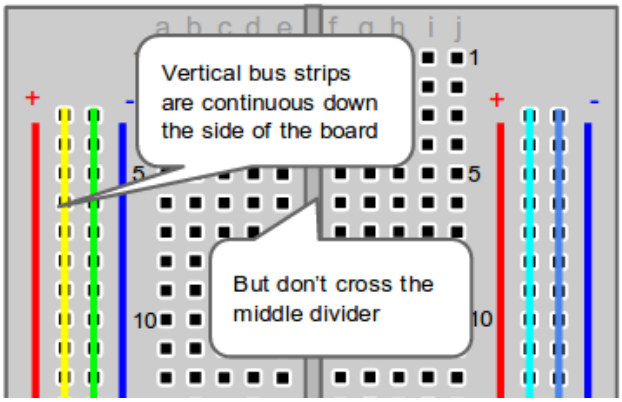
Although it is out of the scope of this outreach day, it is a good idea for you to get a basic knowledge of how electricity works and the physical properties that govern the movement of charge through wires. http://www.instructables.com/id/Basic-Electronics/ is a great website that can help you get started.

The three main concepts that you need for today are to understand that:

1) In order to work, circuits need to be powered

Just like water won't move unless it is on a hill, charge won't move unless there is a potential difference. Our ardunio has the ability to power devices that require both 3.3V and 5V sources.To do this, we will use the vertical bus strips on the breadboard to easily connect to our power source. Since our heart rate module requires a 3.3V source, we will use one jumper wire to connect the 3.3V power source on the arduino to the + vertical bus and another jumper wire to connect ground from the ardunio (GND) to the – vertical bus. This will provide the power we need for our circuit.

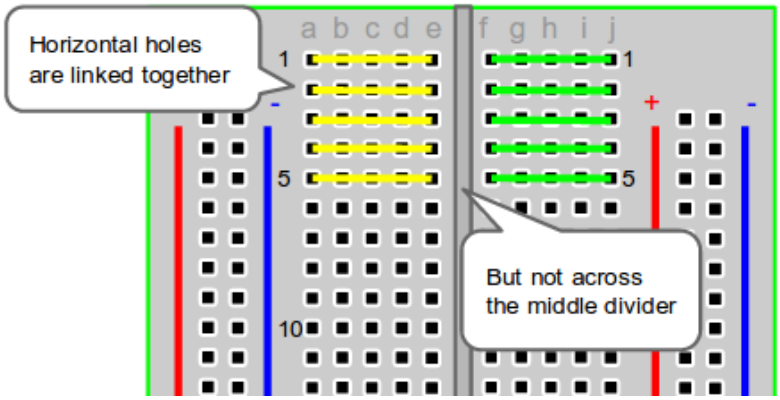
In the example we will do today, this is technically unnecessary, we could connect directly to the ardunio to the sensor with one jumper wire. But it is good form if you decide to add more sensors in the future!



2) In order to work, circuits need to be closed

The jumper wires and the breadboard allow us to close circuits. The breadboard has plates that run along the numbered rows but do not cross the middle divider. This allows us to use jumper wires to make connections from the ardunio, to the heart rate sensor module, without *directly* connecting a wire to each.

If you are having a problem, it is likely that you have an open circuit.



3) Do not create a short circuit

DON'T DO THIS, but if you connect a wire directly from the positive to the negative side of a power supply, you'll create what is called a short circuit. This is a very bad idea. If in doubt, ask if the connection you are making is a short circuit.

Connection Steps - Hardware

1. Choose a region on the breadboard to put your heart rate monitor. Remember, you do not want to place it so that all of the connections are along the same row, that would be a short circuit! In figure 1, the heart rate monitor is placed at the top of the "e" column of the breadboard. Try putting yours somewhere else.
2. Remember, circuits need power! In this example we are going to use the vertical bus strips on the breadboard. Using a jumper wire, connect the (+) vertical bus strip on the breadboard to the 3.3V power source on the ardunio. Now the entire (+) column is connected to this source
3. Similarly, use a jumper wire to connect the GND source on the ardunio to the (-) vertical bus strip on the breadboard.

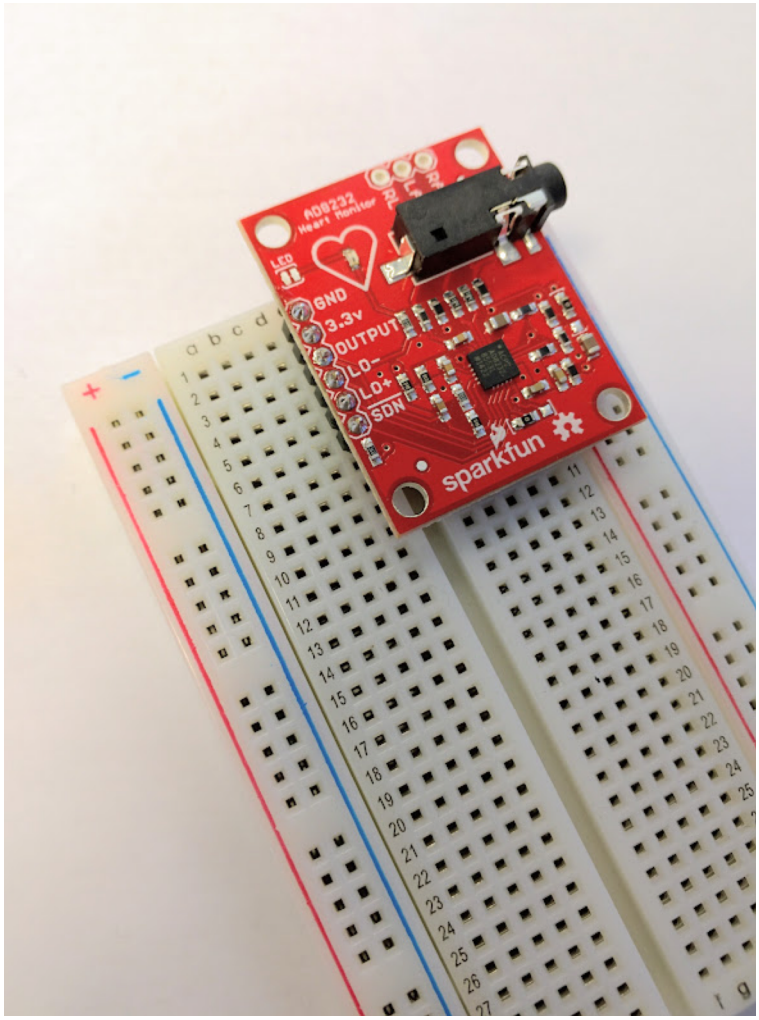


Figure 1: Example of how to place the heart monitor onto the breadboard

4. Now using jumper wires, make the following connections. See figure 2 for help.

Ardunio	**Heart Monitor**
3.3V	3.3V
GND	GND
ANALOG IN - A0	OUTPUT
DIGITAL ~10	LO -
DIGITAL ~11	LO +

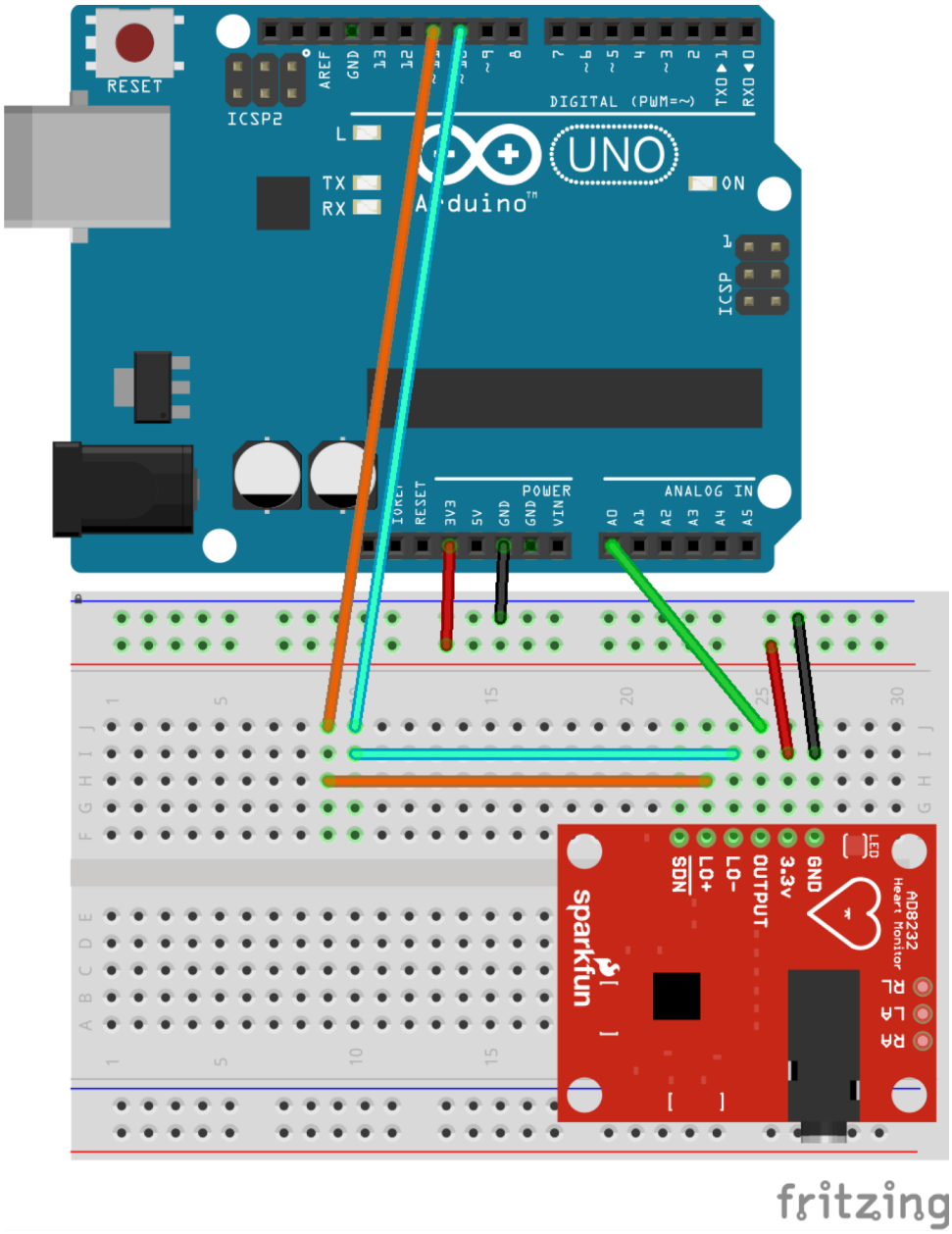


Figure 2: Connection schematic. You do not (and should not) have to make the exact connections that you see above.

We are now connected appropriately but nothing will happen because we need to put software onto the ardunio to tells it how to listen and where to send the data.

Part 2: Programming and Software

Background

You may think computers are smart. After all, we have smartphones, smartTVs, everything seems smart these days! But in reality, computers are exceptionally dumb and obedient machines. They will do *exactly* what we tell them to do. It would be nice (and in the future will probably happen) but if you start shouting commands at your computer in English, it can not understand you. Programming is just writing down commands for the computer in a language it can understand. There are several different programming languages but once you get the hang of one, the rest are just small variations on the same theme.

Today we will be using two different programming languages and integrated development environments (IDE), specifically Ardunio and Processing. An IDE is just a piece of software that helps us write software. It's very meta.

Here are links so you can download the IDE's onto your personal computer:

Arduino Create - https://create.arduino.cc/

Processing IDE 2.2.1 (**NOTE: MUST BE VERSION 2, the code does not currently work with Processing3**) - https://processing.org/download/?processing

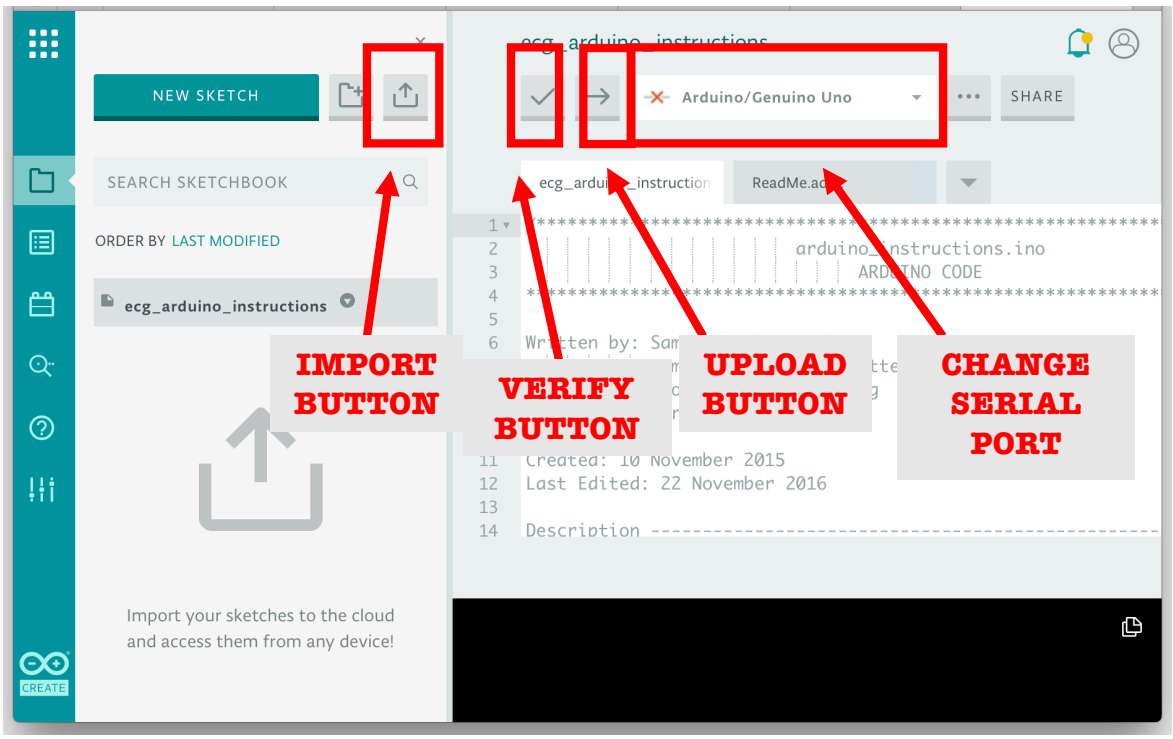
Arduino Create is an online, cloud based, free place to write software for Arduino and I highly recommend it. You will find it is a good tool for your future programming projects.

Downloading and Setting up the Software

You are going to be provided most of the software already written for you. A lot of this is based off of software written by the people at www.sparkfun.com.

1. First, make an Arduino Create account. (https://create.arduino.cc/) Unless you have already used this tool in the past, you may have to sign up for an account. The cool thing is that it will store your Arduino programs in the cloud for free!
2. After you get your create account setup, download all software files from Dr. Bechara's github page. Navigate to the link below and click the "Clone or Download Button". When given an option, choose "Download ZIP". Download it to some location on the computer.

   https://github.com/sbechara/mu-diy-ecg
3. Find the folder called "arduino_instructions_broken". In that folder is a file called "arduino_instructions_broken.ino". Use the import button (see picture below) to import this sketch to the Arduino create program.



4. You can see that this is a short program. When you see the two backslashes "//" or a "/*" followed by a "*/" that indicates to the computer that it can skip these lines of code, you will notice that the IDE colors them grey. These are referred to as comments and is for the programmer's help only. It helps you get an idea of what is happening in your code. When you write code in the future comment as much as possible!
5. Click the verify code button. In the ardunio IDE this is a check mark button (see figure 3). This will tell the IDE to check the code for any errors.
6. If there are any errors fix them, in this case the software was written for you and should have a couple errors. In the future, it is always a good idea to check your code before you upload it though!
7. Once you do not have errors, upload the code to the ardunio. Make sure that it is connected to the computer via a microUSB cable and click the circular right arrow button. You should see the "RX" and "TX" lights flash on the arduino. If they do not flash, there must have been a problem. You may
    - Note: The most common problem is the serial port. You may have to change the serial port so that the Arduino IDE knows where to look for the Arduino hardware. To change the serial port, click the "Tools" dropdown menu on the Arduino IDE toolbar, hover your mouse over the port and change it to a different option. Try uploading again. Repeat until you find the correct serial port.
8. If you did not see any errors, we are ready to start reading ECG data!

Part 3: ECG Analysis

Background on Heart Physiology and ECG

Just like human made machines, your muscles are activated via electrical signals. The electrical signals that are used to regulate your heart follow a very specific and repeating pattern. We call the measurement of these signals an Electrocardiogram (ECG). Each part of the signal corresponds to a different physiological event that occurs within the heart (figure 4).

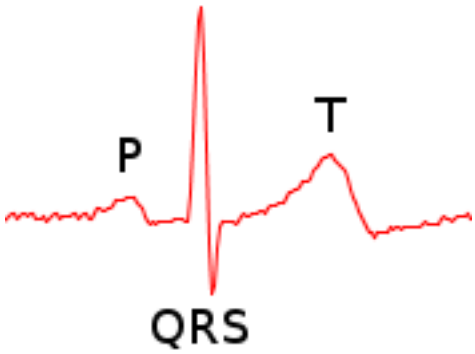


Figure 4: ECG tracing of a single heart beat. Each part of the ECG corresponds to a different physiological event.

Wave	**Heart Physiology**
P-Wave	The first part of the heart beat cycle. The p-wave corresponds to your atria depolarizing and contracting. The atria fill the ventricles up with blood. In essence, the atria are priming the ventricles.
QRS Complex	This is the "thump" of the heart beat. The ventricles rapidly depolarize and powerfully circulate the blood throughout your body. Because your ventricles have a large muscle mass compared to your atria, the peak will have a greater amplitude on the ECG.
T-Wave	You can think of this as your hearts way of re-setting the ventricles. The ventricles repolarize and get ready for the next heart beat. This completes the cardiac cycle.

The electrical signals regulating your heart rhythm are so strong, we can measure them by placing electrodes all the way out on your wrists! Electrodes are just a tool used to interface your body to the computer. They have conducting gel and are specially made to read the electrical signals from your body.

Trained medical professionals can tell a lot by looking at your ECG tracing. They can tell your heart rate by measuring the time between cardiac cycles. They can also diagnose arrhythmias (irregularities in your heart beat) and other life threatening ailments. **Please keep in mind that what we are making *is not a medical or diagnostic grade device and you are most likely not a trained medical professional.* This device is for educational purposes only.**

ECG Acquisition and Analysis

Now that we have successfully uploaded the arduino code and understand a little about heart physiology, we need a way to interface the arduino and computer to read and interpret the data. In this example we are going to use the programming language Processing to read the data from the serial port and to graph it as an ECG.

1. Open up the Processing IDE.
2. As part of the ZIP file you downloaded, there should be a folder called "trace_ecg". Open that folder, and then open the "draw_ekg.pde" file using the Processing 2 IDE. This file includes all of the code necessary to read from the serial port and to create the ECG tracing.
3. **IMPORTANT**: Find the following line in the code:
   myPort = new Serial(this, Serial.list()[0], 9600);
   If you try and run the code and it does not work, it is likely that you need to do is change the value inside of the straight brackets from 0, to a different number that corresponds to the serial port your arduino is connected to (see step 6 from the pervious section). When you try and run the code it will list all of the available serial ports in the display area as an array. The first slot in an arrays start at the 0 position, the next is the 1 position, etc.
4. Read over the rest of the code. It is heavily commented and will give you an idea of what is going on behind the scenes. There are several places that you can customize the graph. More on that later.
5. Make all hardware connections.
   - Make sure that the arduino is plugged into the computer
   - Clean your skin with alcohol swabs where you expect to place an electrode.
   - Connect the electrodes. One on your left wrist, one on your right wrist, and one on your right ankle.
   - Connect the sensor cable to the heart rate monitor.
   - Connect the sensor cable to the electrodes on your skin according to the table below

Blue	Red	Black
Left wrist	Right ankle	Right wrist

6. Press the play button in the top left corner of the Processing IDE. If everything is connected correctly, a new window will pop up and the ECG will start tracing. The output should like similar to figure 5. The program will automatically estimate your heart rate in beats per minute and will label what it thinks is the R-wave of your ECG.
   - For the best results sit still and have your arms resting on the table in front of you.

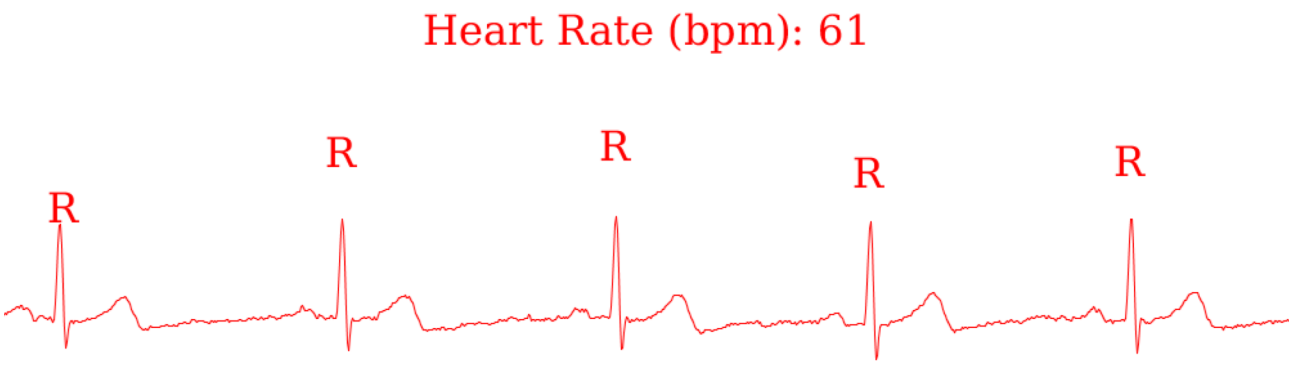


Figure 5: Example of ECG tracing

7. After you get a good tracing and a steady heart rate, you can go into the code and remove the comment in a line of code so you can save the frames of the tracing as PNG files. WARNING: as long as the line is not commented, EVERY SINGLE FRAME of the ECG will be saved. It is recommended that you remove the comment, take a few snapshots of your ECG, close the window or else the files will add up.
8. After you have a few good images of your ECG, disconnect the electrodes from the sensor cable (keep the electrodes on) and go run around to get your heart rate up. Come back and see what has changed about your ECG.
    - *Tip: Do not remove your electrodes until you are totally done using the ECG. They are made with silver and cost about $0.70 per piece!*

Part 5: Hack!

Keep your electrodes on and mess around with the processing code! This is what hackers do (except our reasons are not nefarious). Before you start changing things it is a good idea to save a copy of the working file just in case you break something. It is ok to break things though, that is a part of learning.

If you would like to know more about what each command does, use google to search for Processing then your command of interest. Example google search) Processing stroke

Hacking Ideas

- Changing the color of the ECG tracing by finding a line that looks like this: stroke(0xff, 0, 0);
    - Hint: you can google 0xff to figure out what that means to a computer, then you can google "processing stroke" to see what the processing code does.
    - Hint 2.0: the 0xff is the R of the RGB
- Change the text at the top to be your name's heart rate. Ex) Sam's Heart Rate
- Change the background color
    - Does it change permanently? You might have to change it twice…

- Change the color of *only* the R-wave section of the graph.

- Change the algorithm that detects R-waves. The algorithm that was developed is neither elegant or sophisticated. See if you can adapt it

 Indicates difficulty

Remember that these are just ideas, try something unique and new or make your own program!

Part 6: Discussion Questions

1. Which one of the following statements best answers the question: "Why is it necessary to clean the skin and to use electrically conductive gel?"

   (a) To make sure there is a good electrical connection.

   (b) The gel makes the patient feel less discomfort because it is slimy.

   (c) Cleaning the skin minimizes the possibility of transferring germs to the electrodes.

2. What happened to the ECG signal when the electrodes were tapped during the recording? Why do you think this happens? Sketch an example ECG tracing of this event.

3. What happened to the recorded ECG signal when the volunteer forcefully contracted their forearm muscles? What do you think caused this? Sketch an example ECG tracing of this event.

4. What happened to the heart rate after the volunteer exercised? What do you think caused this? When answering this question, be specific and use what you know about cardiovascular physiology.