Medical Technology

SESSION 3

Building Our Own Medical Device

05 October 2017

Researchers leading this session:

Joanna Brunker

Tom Else

This hand-out belongs to:

Medical devices for Arduino

There are five different kits that we will be working with to measure different properties of the body.

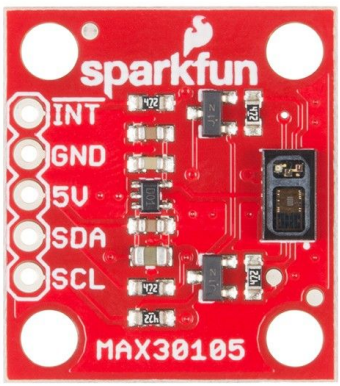
(1) Pulse sensor

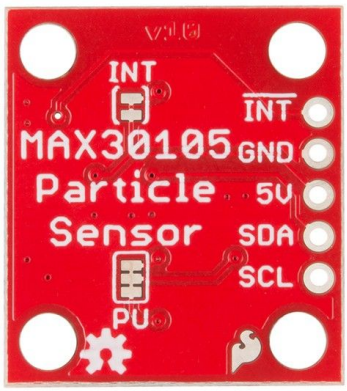
Simply clip the Pulse Sensor to your earlobe or finger tip and plug it into your Arduino and you're ready to read heart rate!

https://www.coolcomponents.co.uk/en/pulse-sensor.html

(2) Particle sensor

The MAX30105 Particle Sensor is a flexible, powerful sensor enabling sensing of distance, heart rate, particle detection and even the blinking of an eye. The MAX30105 has been equipped with three LEDs as well as a very sensitive photon detector. The idea is to pulse the different LEDs, then detect what shines back. Based on the reflected signature it's possible to detect different types of particles or materials (such as oxygenated blood or smoke from a fire).





https://learn.sparkfun.com/tutorials/max30105-particle-and-pulse-ox-sensor-hookup-guide?_ga=1.97228895.1597862514.1479216075

Medical devices for Arduino

(3) Heart rate monitor

The AD8232 Heart Rate Monitor is a neat little chip used to measure the electrical activity of the heart. This electrical activity can be charted as an ECG or Electrocardiogram, simply by connecting to the Biomedical Sensor Pads and the Sensor Cable - Electrode Pads (3 connector) to create an ECG monitor.

Electrocardiography is used to help diagnose various heart conditions.

https://learn.sparkfun.com/tutorials/ad8232-heart-rate-monitor-hookup-guide

(4) EMG (Electromyography) sensor

The EMG detector is a bridge between your muscles and microcontrollers, the sensor gathers small muscle signals and then amplify and filters them to produce an output signal that can be recognized.

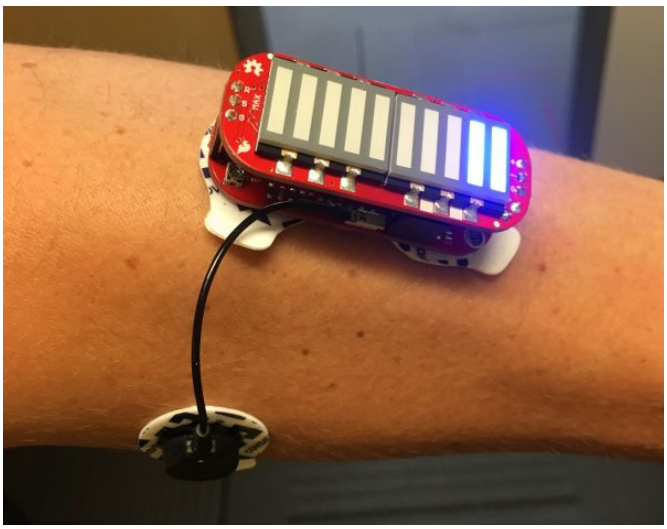
https://www.coolcomponents.co.uk/en/emg-detector.html

Medical devices for Arduino

(5) MyoWare Muscle Sensor Development Kit

Using our muscles to control things is the way that most of us are accustomed to doing it. We push buttons, pull levers, move joysticks…but what if we could take the buttons, levers and joysticks out of the equation? This is the MyoWare Muscle Sensor Development Kit, an Arduino-powered, electromyography (EMG) sensor kit that provides you with a MyoWare Muscle Sensor, each MyoWare shield developed at SparkFun, and everything you need to connect them all together.

The MyoWare board included in this kit acts by measuring the filtered and rectified electrical activity of a muscle; outputting 0-Vs Volts depending on the amount of activity in the selected muscle, where Vs signifies the voltage of the power source. It's very simple to hook up this kit, attach shields to the muscle sensor, stick on a few included electrodes, read the voltage out and flex some muscles!



https://learn.sparkfun.com/tutorials/myoware-muscle-sensor-kit?_ga=2.39886562.1973069348.1503480036-1575493297.1484668914

https://cdn.sparkfun.com/datasheets/Sensors/Biometric/MyowareUserManualAT-04-001.pdf

Operating a Pulse Sensor

Introduction:

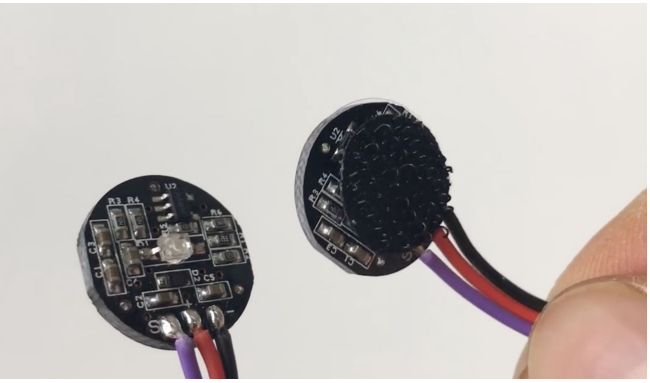
Heart rate data can be really useful whether you're designing an exercise routine, studying your activity or anxiety levels or just want your shirt to blink with your heart beat. The problem is that heart rate can be difficult to measure. Good news… the Pulse Sensor can solve that problem!

The Pulse Sensor is a plug-and-play heart-rate sensor for Arduino. It can be used by students, artists, athletes, makers, and game & mobile developers who want to easily incorporate live heart-rate data into their projects.It essentially combines a simple optical heart rate sensor with amplification and noise cancellation circuitry making it fast and easy to get reliable pulse readings. Also, it sips power with just 4mA current draw at 5V so it's great for mobile applications.

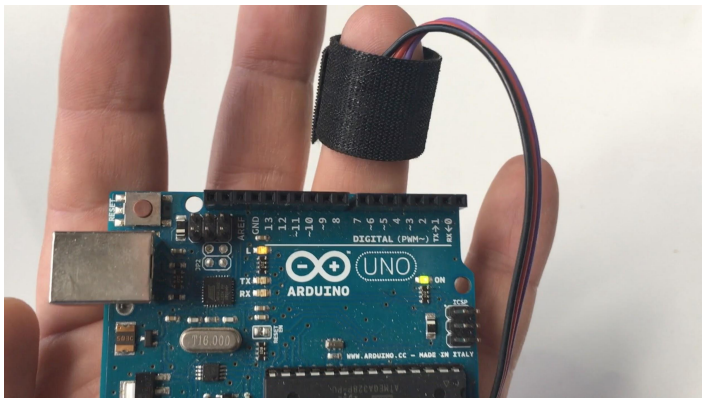
Simply clip the Pulse Sensor to your earlobe or finger tip and plug it into your 3 or 5 Volt Arduino and you're ready to read heart rate! The 24" cable on the Pulse Sensor is terminated with standard male headers so there's no soldering required. Of course Arduino example code is available as well as a Processing sketch for visualizing heart rate data.

1. First stick the velcro stickers on the back of the board

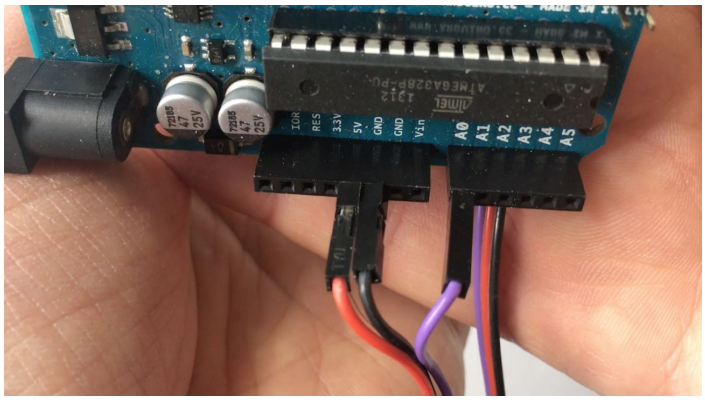
2. Then stick the vinyl dot on the front



3. You can now put it on the velcro finger strap



4. Plug the pulse sensor power and ground into the power strip on the Arduino
5. Plug the purple cable into the Analog zero pin (A0)

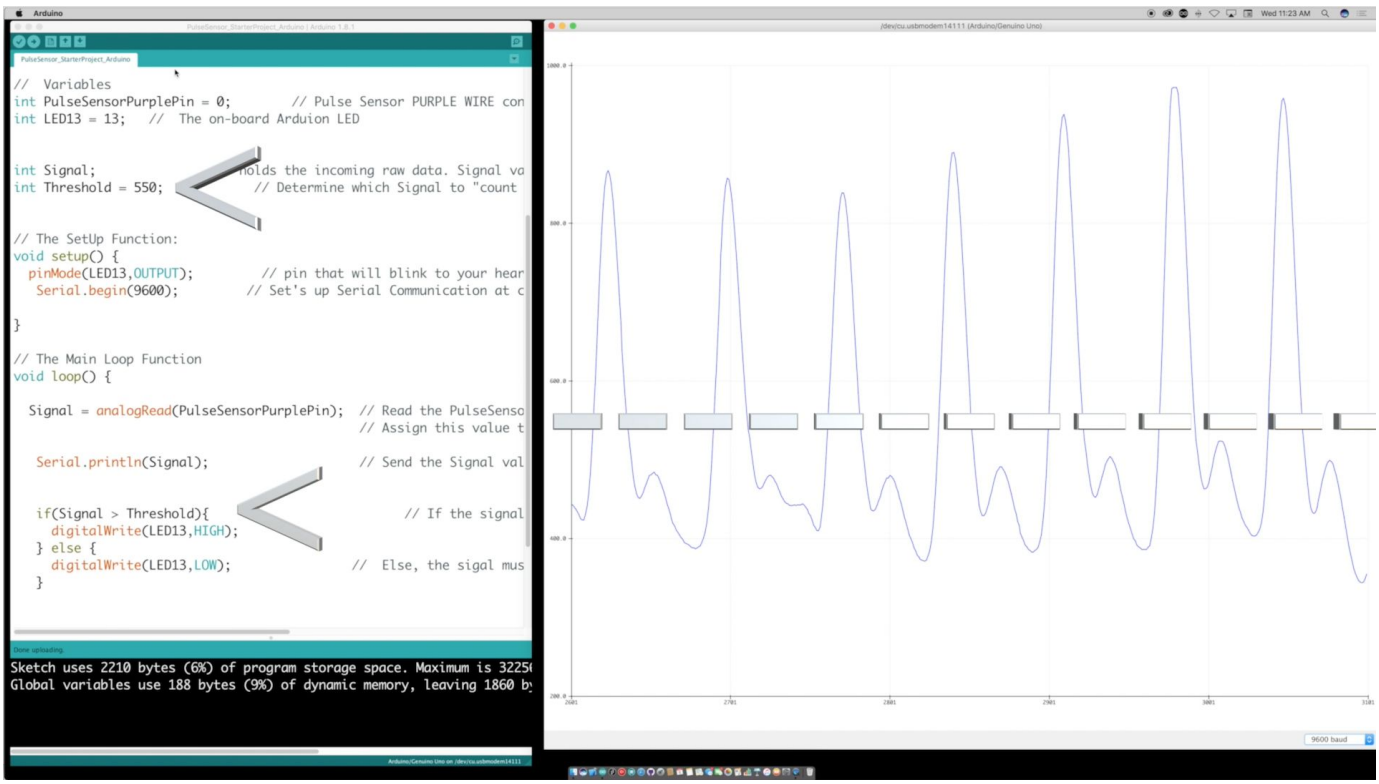


Open Up the Arduino IDE software on your computer.
Coding in the Arduino language will control your circuit.

Go to the webpage:
https://github.com/WorldFamousElectronics/PulseSensorStarterProject/tree/master/PulseSensor_StarterProject_Arduino

Open the .ino file in the Arduino Software.

You should see your pulse in real time on the serial plotter, built from the output from the purple pin in A0.



The threshold in the code sets what you consider to be your heartbeat. Every time the pulse sensor signal passes over the threshold we blink the LED on pin 13.

Extension:

Can you adapt the code so that it reads out your heart rate?

The code on the next page gives an example of how you might do this...

```
//  Variables
int PulseSensorPurplePin = 0;        // Pulse Sensor PURPLE WIRE connected to ANALOG PIN 0
int LED13 = 13;   //  The on-board Arduion LED

unsigned long beatRecordedatMs[4] = {0,0,0,0} ;
int Signal;                // holds the incoming raw data. Signal value can range from 0-1024
int Threshold = 550;            // Determine which Signal to "count as a beat", and which to ingore.
int wasBeat = 0;
// The SetUp Function:
void setup() {
  pinMode(LED13,OUTPUT);         // pin that will blink to your heartbeat!
   Serial.begin(9600);         // Set's up Serial Communication at certain speed.

}

// The Main Loop Function
void loop() {

  Signal = analogRead(PulseSensorPurplePin);  // Read the PulseSensor's value.
                                              // Assign this value to the "Signal" variable.

   //Serial.println(Signal);                    // Send the Signal value to Serial Plotter.


   if(Signal > Threshold){// If the signal is above "550", then "turn-on" Arduino's on-Board LED.
     digitalWrite(LED13,HIGH);
     wasBeat = 1;
   } else {
    if(wasBeat == 1)
    {
      beatRecordedatMs[3] = beatRecordedatMs[2] ;
      beatRecordedatMs[2] = beatRecordedatMs[1] ;
      beatRecordedatMs[1] = beatRecordedatMs[0] ;
      beatRecordedatMs[0] = millis() ;
      int elapsedTimeof4Pulses = beatRecordedatMs[0] - beatRecordedatMs[3] ;
      float pulseRateBeatsPerMinute =  ( elapsedTimeof4Pulses / 4.0 ) * 60.0  / 1000.0;
      Serial.println(pulseRateBeatsPerMinute);
      wasBeat = 0;
    }
     digitalWrite(LED13,LOW);                //  Else, the sigal must be below "550", so "turn-off" this LED.
   }


delay(10);


}
```

Notes