Team Quadcopter: Code

Arduino Code

```
// Description: This is an Arduino program to measure the value of a resistor
in ohms.

// Include bluetooth libraries
#include <SPI.h>
#include <boards.h>
#include <RBL_nRF8001.h>
#include <services.h>

// Initialize global variables
int vinSupplier= A0; // The 5V input voltage pin
int vinReader = A2; // The pin that reads the exact input voltage
int voutReader = A1; // The pin that reads the voltage accross the resistor
being measured
int triggerPin = A3; // The pin that listens to whether the button used to
initiate the measurement is pressed
int ledIndicator = A4; // Pin used to drive LED to alert the user a measurement
is in progress
int voutRaw= 0; // A variable to hold the ADC measurement of the voltage
accross measured resistor
int vinRaw = 0; // A variable to hold the ADC measurement of the input voltage
float Vin= 0; // vinRaw mapped to an actual voltage value
float Vout= 0; // voutRaw mapped to an actual voltage value
float Rknown= 14840;//14840;//991; // the value of the known resistor
float Resistance= 0; // the calculation of resistance
char charRes[10]; // The character array representation of the resistance for
ble transmission
int triggerPinVoltage = 0; // the ADC measurement of the switch voltage

boolean inprogress = false; // a boolean to let us know whether a test is
currently in progress
boolean canceled = false;
int avgLength = 1000; // The number of measurements to be averaged
float average(float a[]); // An averaging function

void setup()
{
    // Set the vinSupplier pin to be an output
    pinMode(vinSupplier, OUTPUT);
    pinMode(ledIndicator, OUTPUT);
```

```
    // Begin serial communication
    Serial.begin(9600);

    // Init. and start BLE library.
    ble_begin();
}

void loop()
{
    // Check to see if the push button is pressed to initiate a measurement
    triggerPinVoltage = analogRead(triggerPin);

    // Only measure if we have received a ble command to do s0
    if( ble_available() ){
      analogWrite(ledIndicator, 255);
      Serial.println("Ble Available");

      // Exhasut the ble data buffer so we have nothing left in there to cause
a second test to run without user prompt
      while ( ble_available() )
        ble_read();

      // Let's take a whole bunch of measurements and average
      // them to increase precision
      //float resistanceArray[avgLength];
      float sum = 0;
      int i;
      for(i=0; i<avgLength && canceled == false; i++){

        if(ble_available()){
          canceled = true;
          Serial.println("test canceled");
          while ( ble_available() )
            ble_read();
        }
        ble_do_events();
        // Set inprogress to true to indicate a test has begun due to
        // a button press
        inprogress = true;

        // Drive the input to the ohm meter with a 5V input
        analogWrite(vinSupplier, 255);

        // Read the voltage across the measured resistor
        voutRaw= analogRead(voutReader);

        // Read the voltage being supplied by vinSupplier
        vinRaw = analogRead(vinReader);
```

```
   // Turn off the input voltage
  analogWrite(vinSupplier, 0);

  // Map the ADC measured voltages to actual voltage values
  Vin = (vinRaw/1024.0)*5.0;
  Vout = (voutRaw/1024.0)*5.0;

  // Compute the current flowing through the known resistor
  float current = (Vin - Vout)/Rknown;

  // Determine the resistance of the resistor we are measuring
  Resistance = Vout/current;
  sum = sum + Resistance;
  Serial.println(Resistance);
  //Serial.println(',');

   // Turn off the input voltage
  //analogWrite(vinSupplier, 0);

  //Add the resistance value to our averaging array
  //resistanceArray[i] = Resistance;
  //delay(1000);



}// End for loop

canceled = false;

//float finalResistance = average(resistanceArray);
float finalResistance2 = sum/avgLength;
Serial.println("Final Resistance");
// Serial.println(finalResistance);
Serial.println(finalResistance2);

// Convert float resistance to char array so that we
// can transmit over ble
dtostrf(finalResistance2, 4, 3, charRes);

for(int i=0;i<sizeof(charRes);i++)
{
  // Write to ble and to console
  ble_write(charRes[i]);
  Serial.print(charRes[i]);
}

ble_write('\n'); //the new line character let's the phone know the
```

```
message is over
      analogWrite(ledIndicator, 0);

    }
    else if(triggerPinVoltage > 10 && inprogress){
      // Do not do anything if the button has remained depressed since
      // the latest resistance measurement
      //Serial.println("Idling");

    }
    else{
      // This means the button has be unpressed so we can finally
      // say this resistance measurement is over and can start a new one
      // once the button is pressed again.
      inprogress = false;
    }

    // Communicate changes
    ble_do_events();
    //delay(1000);


}

float average(float a[]){
     int i;
     float avg, sum=0.0;
     for(i=0;i<avgLength;++i){
       sum+=a[i];
     }
     avg =(sum/avgLength);
     return avg;
}
```

App Code

```
angular.module('starter', ['ionic'])

.run(function($ionicPlatform, $templateCache, $rootScope, $interval, $timeout)
{

  // Load a random value to populate the knob
  $rootScope.currentValue = 41;
```

```
  $rootScope.weight = 0;

  // Create a global ble connected status variable for use with ng-show/ng-hide
  $rootScope.connected = false;

  $ionicPlatform.ready(function() {

    // Hide the accessory bar by default (remove this to show the accessory bar
above the keyboard
    // for form inputs)
    if(window.cordova && window.cordova.plugins.Keyboard) {
      cordova.plugins.Keyboard.hideKeyboardAccessoryBar(true);
    }
    if(window.StatusBar) {
      StatusBar.styleDefault();
    }

    // monitor ble connectivity status every second
    $rootScope.monitorConnectivity = function(){
      $interval(function(){
        bluetoothSerial.isConnected(
            function() {
                console.log("bluetooth connected");
                $rootScope.connected = true;
            },
            function() {
                $rootScope.connected = false;

                // If we are not already trying to connect, attempt to
reconnect
                if(!$rootScope.connecting){

                  $timeout(function(){
                    console.log("attempting to RECONNECT")
                    $rootScope.connecting = true;
                    $rootScope.initiateConnection();

                  }, 1000)

                }
                console.log("Bluetooth is *not* connected");
            }
        );
      }, 1000)
    }();

    $rootScope.initiateConnection = function(){
      console.log("running initiateConnection..")
```

```
      // Let's scan the environment for a ble device with the name "BLE Shield"
      bluetoothSerial.list(function(devices) {
          console.log("in success callback")

          if(devices.length === 0)
            return $rootScope.connecting = false;

          var bleShield = _.findWhere(devices, {name: "BLE Shield"});

          // JSON output of bleShield looks like this:
          //[{"id":"55174456-779D-D60E-82D4-EA927560790C","name":"BLE
Shield","uuid":"55174456-779D-D60E-82D4-EA927560790C"}]

          // Now that we have found the bleShield and have its id, lets
connect!
          bluetoothSerial.connect(bleShield.uuid, $rootScope.connnectSuccess,
function(){$rootScope.connecting = false;});
      }, function(){
        console.log("in fail callback")
      });

    }

  $rootScope.gotMessage = function(data){
    // This function gets called when we receive the resistance measurement
from the arduino
    console.log("got gotMessage called")
    console.log(data)
    $rootScope.newResistance = parseFloat(data);
  }

  $rootScope.connnectSuccess = function (){
    console.log("We connected via bluetooth");
    console.log("attempting to subscribez")
    $rootScope.connecting = false;
    // $rootScope.connected = true;
    // $scope.connected = true;
    console.log("connected: " + $rootScope.connected)

    bluetoothSerial.subscribe("\n", $rootScope.gotMessage, function(){
      console.log("got an error..");
    });
  }


  });
})
```

```
.config(function($stateProvider, $urlRouterProvider) {
  $urlRouterProvider.otherwise('/')

  $stateProvider
    .state('home', {
      url: '/',
      //template: '<p style="color: white;">Hello, world!</p>'
      templateUrl: 'home.html',
      controller: 'homeCtrl'
    })

    .state('enterWeight', {
      url: '/enterWeight',
      templateUrl: 'enter-weight.html',
      controller: 'enterWeightCtrl'
    })

    .state('measuring', {
      url: '/measuring',
      templateUrl: 'measuring.html',
      controller: 'measuringCtrl'
    })
})

.controller('homeCtrl', function($scope, $rootScope, $timeout) {

  $timeout(function(){

    if(!$rootScope.connected)
      $rootScope.initiateConnection();

  }, 1000)


  function connectFailure(){
    console.log("Could not connect");
  }

      function gotMessageError(error){
    // If there is an error receiving the resistance measurement from the
arduino, this function
    // gets called
            console.log("error")
      }

  $timeout(function(){

    // Run the code to initiate our body fat percentage knob (this should
```

```
ideally be within a directive)
    initiateKnob();
    initiateLoader($rootScope.currentValue);

    // Set knobInitiated to true so we can unhide the now not-so-ugly knob
    $scope.knobInitiated = true;
  })


})

.controller('enterWeightCtrl', function($scope, $rootScope, $timeout,
$interval, $state) {
  $scope.fullWeight = 0;

  $scope.storeWeight = function(){
    $rootScope.weight = parseFloat($scope.fullWeight);
    $state.go("measuring");
  }
  //console.log($rootScope.currentValue)
})

.controller('measuringCtrl', function($scope, $rootScope, $timeout, $interval,
$state) {
  console.log($rootScope.currentValue)

  delete $rootScope.newResistance;

  bluetoothSerial.write("start_test", function(){
      console.log("started_test");
    }, function(){
      console.log("failed to start_test");
    });

  $scope.progressValue = 0;

  incrementProgress = $interval(function(){
    console.log("interval called")
    if($scope.progressValue < 99 && typeof $rootScope.newResistance ===
"undefined")
      $scope.progressValue++;

    else if(typeof $rootScope.newResistance === "undefined"){
      // wait to get resistance
    }
    else {
      $scope.progressValue = 100;
      // cancel the interval since it appears to be global accross all
```

```
controllers
      $interval.cancel(incrementProgress);

      console.log("about to compute regression")
      // Define our leanMass regression coefficients
      // var B0 = -13.36117;
      // var B1 = 0.0085998;
      // var B2 = 0.5964701;

      var B0 = 1.207298;
      var B1 = -0.0047503;
      var B2 = 0.5542703 ;

      // Define our waterMass regression coefficients
      var C0 = 14.35698;
      var C1 = -0.0079816;
      var C2 = 0.3226745;

      //$rootScope.newResistance = 600;
      console.log("$rootScope.weight: " + $rootScope.weight);
      //$rootScope.weight = 130;
      // Let's compute the regressionjquery.classyloader.js
      var leanMass = B0 + B1*$rootScope.newResistance + B2*$rootScope.weight;
      console.log("leanMass: " + leanMass)
      var waterMass = 0.35*$rootScope.weight;//C0 + C1*$rootScope.newResistance
+ C2*$rootScope.weight;
      console.log("waterMass: " + waterMass)
      var fatMass = $rootScope.weight - leanMass - waterMass;
      console.log("fatMass: " + fatMass)

      // set a random value for the new body fat percentage from (0-100)
      $rootScope.currentValue = Math.round(100*(fatMass/($rootScope.weight -
waterMass))); //Math.floor(Math.random() * 100) + 1;
      //$rootScope.currentValue = 21.9;
      console.log("fatPercentage: " + $rootScope.currentValue);
      delete $rootScope.newResistance;
      // Go back to the home state to display the new bodyfat percentage
      $state.go('home');
    }
  }, 80)

  $scope.cancelMeasurement = function(){
    $interval.cancel(incrementProgress);
    console.log("cancelMeasurement called")
    bluetoothSerial.write("cancel_test", function(){
      console.log("canceled test");
      $state.go('home');
    }, function(){
```

```
      console.log("failed to cancel_test");
    });

  }

})

function initiateKnob(){

  // This function is used to initialize the jQuery knob used to display the
body fat percentage
  // to the user in the home state

   $(function($) {
               $(".knob").knob({
                    change : function (value) {
                        //console.log("change : " + value);
                    },
                    release : function (value) {
                        //console.log(this.$.attr('value'));
                        console.log("release : " + value);
                    },
                    cancel : function () {
                        console.log("cancel : ", this);
                    },
                    /*format : function (value) {
                        return value + '%';
                    },*/
                    draw : function () {
                        // "tron" case
                        if(this.$.data('skin') == 'tron') {
                            this.cursorExt = 0.3;
                            var a = this.arc(this.cv)  // Arc
                                , pa                   // Previous arc
                                , r = 1;
                            this.g.lineWidth = this.lineWidth;
                            if (this.o.displayPrevious) {
                                pa = this.arc(this.v);
                                this.g.beginPath();
                                this.g.strokeStyle = this.pColor;
                                this.g.arc(this.xy, this.xy, this.radius -
this.lineWidth, pa.s, pa.e, pa.d);
                                this.g.stroke();
                            }
                            this.g.beginPath();
                            this.g.strokeStyle = r ? this.o.fgColor :
this.fgColor ;
                            this.g.arc(this.xy, this.xy, this.radius -
```

```
this.lineWidth, a.s, a.e, a.d);
                            this.g.stroke();
                            this.g.lineWidth = 2;
                            this.g.beginPath();
                            this.g.strokeStyle = this.o.fgColor;
                            this.g.arc( this.xy, this.xy, this.radius -
this.lineWidth + 1 + this.lineWidth * 2 / 3, 0, 2 * Math.PI, false);
                            this.g.stroke();
                            return false;
                        }
                    }
                });
                // Example of infinite knob, iPod click wheel
                var v, up=0,down=0,i=0
                    ,$idir = $("div.idir")
                    ,$ival = $("div.ival")
                    ,incr = function() { i++; $idir.show().html("+").fadeOut();
$ival.html(i); }
                    ,decr = function() { i--; $idir.show().html("-").fadeOut();
$ival.html(i); };
                $("input.infinite").knob(
                                {
                                min : 0
                                , max : 20
                                , stopper : false
                                , change : function () {
                                                if(v > this.cv){
                                                    if(up){
                                                        decr();
                                                        up=0;
                                                    }else{up=1;down=0;}
                                                } else {
                                                    if(v < this.cv){
                                                        if(down){
                                                            incr();
                                                            down=0;
                                                        }else{down=1;up=0;}
                                                    }
                                                }
                                                v = this.cv;
                                            }
                                });
            });

}

function initiateLoader(pct){
```

```
$('.loader').ClassyLoader({
    speed: 50,
    diameter: 80,
    fontSize: '30px',
    fontFamily: 'Arial',
    fontColor: 'rgb(53, 188, 228)',
    lineColor: 'rgb(53, 188, 228)',
    remainingLineColor: 'rgba(73, 125, 164, 0.1)',
    percentage: pct,
    lineWidth: 20,
    start: 'top',
    //remainingLineColor: 'rgba(200,200,200,0.1)'
});
}
```