Analog EMG Sensor by OYMotion SKU:SEN0240

Introduction

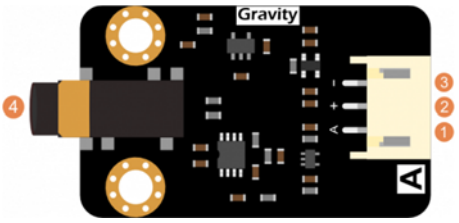
This EMG sensor is launched by DFRobot and OYMotion cooperation, which can reflect muscle and neural activities of human by detecting sEMG.
This sensor integrates filtering circuit and amplified circuit. It amplifies minimal sEMG within ±1.5mV 1000 times and depresses noises (especially power frequency interference) by differential input and analog filter circuit. The output signal is analog, which takes 1.5V as the reference voltage. The output voltage range is 0~3.0V. The signal strength is depends on muscle activities. The output signal waveform indicates the muscle activity and helps to analyse and research the sEMG. Specifically, we can use Arduino as a controller to detect muscle activities, e.g. check whether the muscle is tense; the muscle strength;etc.
This is an active induction sensor can provide high quality signal collection and it is easy to use. Simple preparations are needed to apply the module to whether static or dynamic areas. Dry electrode is applied to the module and good quality signal are available even without conductive gel. Compared with disposable conductive gel needed by medical electrodes, it is more convenient to use and has longer service life. Therefore, it is more suited to common users.
Measurements with an analog EMG sensor are noninvasive, convenient and this can be applied to human-computer interactions. With the development of microcontrollers and integrated electric circuits, EMG circuits and sensors have not just been applied to traditional medical muscle detection researches but control systems.

The supply voltage range is 3.3~5.5V; The supply current should not be less than 20mA; The ripple current and disturbance current should be as low as possible. Stabilized DC voltage is recommended.
The effective spectrum range is 20Hz～500Hz, and the ADC converter which has higher than 8-bit resolution and 1 KHz frequency are recommended to take samples and digitized to keep original information.
Placing the metal dry electrode should consistent with the direction of muscle.
The product is not a professional medical device and cannot diagnose and cure disease as an assistant device.

Specification

- **Signal Transmitter Board**
  Supply Voltage: +3.3V～5.5V
  Operating Voltage: +3.0V
  Detection Range: +/-1.5mV
  Electrode Connector: PJ-342
  Module Connector: PH2.0-3P
  Output Voltage: 0～3.0V
  Operating Temperature: 0～50℃
  Size: 22mm*35mm (0.87inch*1.38inch)
- **Dry Electrode Board**
  Electrode Connector: PJ-342
  Wire Length: 50cm(19.69inch)
  Plate Size: 22 * 35 mm(0.87inch*1.38inch)
  weight: 36g

Board Overview



Analog EMG Sensor by OYMotion

Num	Label	Description
1	A	Analog Signal Output(0~3.0V)
2	+	Power Supply Anode(3.3~5.5V)
3	-	Power Supply Cathode(0V)
4	PJ-342	Probe Wiring Connector

Tutorial

The tutorial will show you how to use the analog EMG sensor by printing electromyogram waveform with Serial Plotter of Arduino IDE.

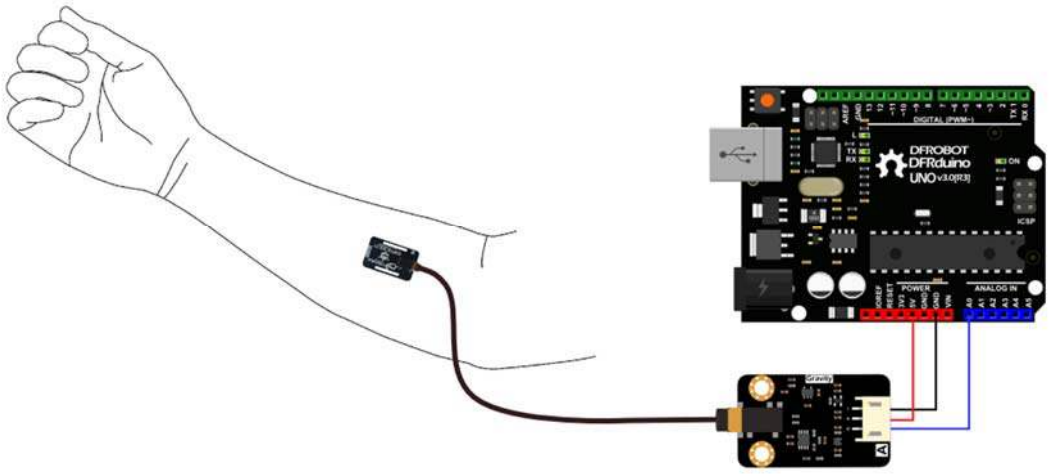
Requirements

- **Hardware**
  DFRduino UNO R3 (or similar) x 1
  Signal Transmitter Board x1
  Dry Electrode Plate x1
  Dry Electrode Cable x1
  Gravity Analog Cable 3Pin（or Dupont Wires） x1

- **Software**
  Arduino IDE (Version requirements: V1.0.x or V1.8.x), Click to Download Arduino IDE from Arduino®

  https://www.arduino.cc/en/Main/Software%7C

Connection Diagram



Sample Code

This sample code needs EMGFilters library file, please download and install the EMGFilters Library file.
How to install Libraries in Arduino IDE https://www.arduino.cc/en/Main/Software%7C

/*
* Copyright 2017, OYMotion Inc.
* All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
*
* 1. Redistributions of source code must retain the above copyright
*   notice, this list of conditions and the following disclaimer.
*
* 2. Redistributions in binary form must reproduce the above copyright
*   notice, this list of conditions and the following disclaimer in
*   the documentation and/or other materials provided with the
*   distribution.
*
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
* "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
* LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
* FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
* COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
* INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
* BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS
* OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED
* AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
* OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF
* THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH
* DAMAGE.
*
*/

```
#if defined(ARDUINO) && ARDUINO >= 100
#include "Arduino.h"
#else
```

```
#include "WProgram.h"
#endif

#include "EMGFilters.h"

#define TIMING_DEBUG 1

#define SensorInputPin A0 // input pin number

EMGFilters myFilter;
// discrete filters must works with fixed sample frequence
// our emg filter only support "SAMPLE_FREQ_500HZ" or "SAMPLE_FREQ_1000HZ"
// other sampleRate inputs will bypass all the EMG_FILTER
int sampleRate = SAMPLE_FREQ_1000HZ;
// For countries where power transmission is at 50 Hz
// For countries where power transmission is at 60 Hz, need to change to
// "NOTCH_FREQ_60HZ"
// our emg filter only support 50Hz and 60Hz input
// other inputs will bypass all the EMG_FILTER
int humFreq = NOTCH_FREQ_50HZ;

// Calibration:
// put on the sensors, and release your muscles;
// wait a few seconds, and select the max value as the threshold;
// any value under threshold will be set to zero
static int Threshold = 0;

unsigned long timeStamp;
unsigned long timeBudget;

void setup() {
    /* add setup code here */
    myFilter.init(sampleRate, humFreq, true, true, true);
```

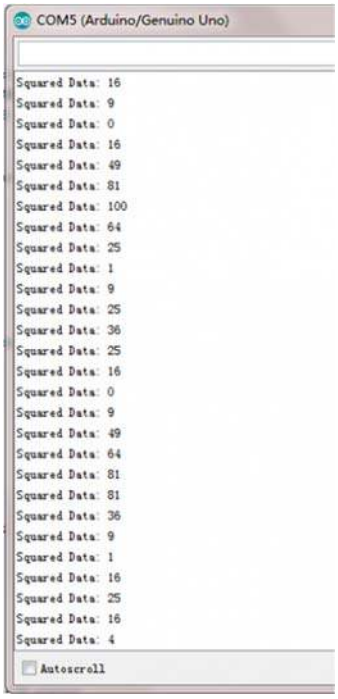
```
    // open serial
    Serial.begin(115200);

    // setup for time cost measure
    // using micros()
    timeBudget = 1e6 / sampleRate;
    // micros will overflow and auto return to zero every 70 minutes
}

void loop() {
    /* add main program code here */
    // In order to make sure the ADC sample frequence on arduino,
    // the time cost should be measured each loop
    /*------------start here-------------------*/
    timeStamp = micros();

    int Value = analogRead(SensorInputPin);

    // filter processing
    int DataAfterFilter = myFilter.update(Value);

    int envlope = sq(DataAfterFilter);
    // any value under threshold will be set to zero
    envlope = (envlope > Threshold) ? envlope : 0;

    timeStamp = micros() - timeStamp;
    if (TIMING_DEBUG) {
        // Serial.print("Read Data: "); Serial.println(Value);
        // Serial.print("Filtered Data: ");Serial.println(DataAfterFilter);
        Serial.print("Squared Data: ");
        Serial.println(envlope);
        // Serial.print("Filters cost time: "); Serial.println(timeStamp);
        // the filter cost average around 520 us
    }
```

```
    /*------------end here---------------------*/
    // if less than timeBudget, then you still have (timeBudget - timeStamp)
to
    // do your work
    delayMicroseconds(500);
    // if more than timeBudget, the sample rate need to reduce to
    // SAMPLE_FREQ_500HZ
}
```

Calibration

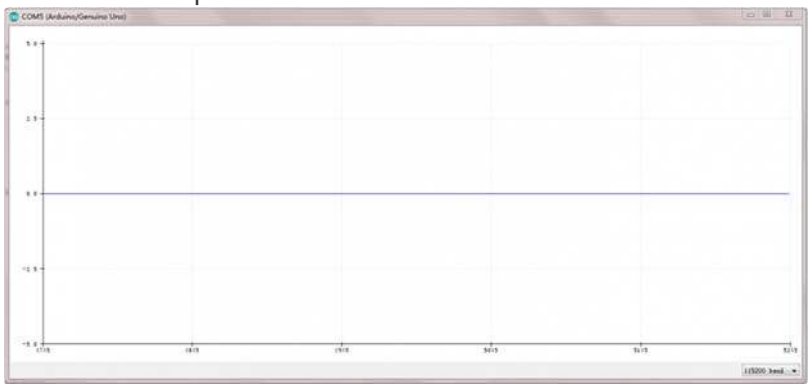
Calibration is recommended for every time when you use it, for the EMG signal is different in different position even of the same person.

- 1. Modify the variable **Threshold** in the sample code to 0, namely: **static int Threshold = 0;**
- 2. Upload the sample code to Arduino control board and open the serial monitor of Arduino IDE to check the printed value.
- 3. Relax the arm muscle and check the printed value of serial port. Calm down and relax for a while to check and record the maximum value printed by the serial monitor. If the value is too large, such as more than 1000, you can try fine-tune position of the dry electrode.

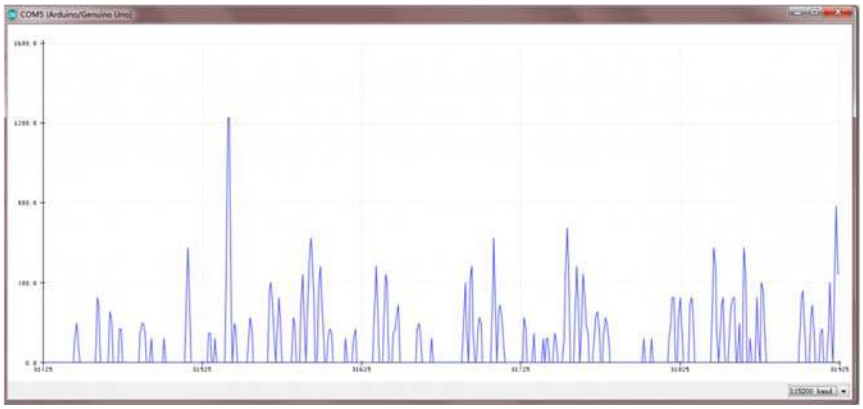


- 4. Modify the variable **Threshold** in the sample code to the recorded maximum value, and reload the sample code to Arduino control board.
- 5. Open the serial plotter of arduino IDE and the myoelectricity waveform could be seen.

In general, the waveform is a line of 0 when the muscle is relaxed, shown as below. It is normal along with occasional peak waveform.



Obviously myoelectricity waveform could been seen in the process of making a fist, shown as below.



Basic Application

This application explains the way to distinguish whether the EMG signal is made by fisting counts. It can be applied to push-up counts, dumbbell curl counts. It also can be applied to human-computer interaction, such as play Flappy Bird with EMG signal made by fist.
Please refer to the Chapter 4: 4.1 Preparation and 4.2 Connection Diagram for hardware connection and sensor position.
This sample code needs EMGFilters library file, please download and install the EMGFilters Library first. https://codeload.github.com/yuyouliang/EMG_Filter/zip/master

How to install Libraries in Arduino IDE https://www.arduino.cc/en/Guide/Libraries#.UxU8mdzF9H0%7C

/*
   Copyright 2017, OYMotion Inc.
   All rights reserved.

   Redistribution and use in source and binary forms, with or without
   modification, are permitted provided that the following conditions
   are met:
   1. Redistributions of source code must retain the above copyright
      notice, this list of conditions and the following disclaimer.
   2. Redistributions in binary form must reproduce the above copyright
      notice, this list of conditions and the following disclaimer in
      the documentation and/or other materials provided with the
      distribution.

   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
   "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
   LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
   FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
   COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
   INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
   BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS
   OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED
   AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
   OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF
   THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH
   DAMAGE.

```
*/


#if defined(ARDUINO) && ARDUINO >= 100

#include "Arduino.h"

#else

#include "WProgram.h"

#endif


#include "EMGFilters.h"

#define SensorInputPin A0   //sensor input pin number


/*

  Define the `threshold` variable as 0 to calibrate the baseline val
ue of input sEMG signals first.

  After wiring the sEMG sensors to the Arduino board, wear the sEMG
sensors. Relax your muscles for a few seconds,

  you will be able to see a series of squared sEMG signals values ge
t printed on your serial terminal.

  Choose the maximal one as the baseline by setting the `threshold`
variable. Then rebuild this project.

  The `envelope`, which is the squared sEMG signal data, will be pri
nted to the serial line.

  The developer can plot it using the Arduino SerialPlotter.


  Note:

      After calibration, Any squared value of sEMG sigal below the b
aseline will be treated as zero.

      It is recommended that you do calibration every time you wear
the sEMG sensor.

*/

unsigned long threshold = 0;  // threshold: Relaxed baseline values.
(threshold=0:in the calibration process)

unsigned long EMG_num = 0;      // EMG_num: The number of statistica
l signals


EMGFilters myFilter;

/*

  Set the input frequency.
```

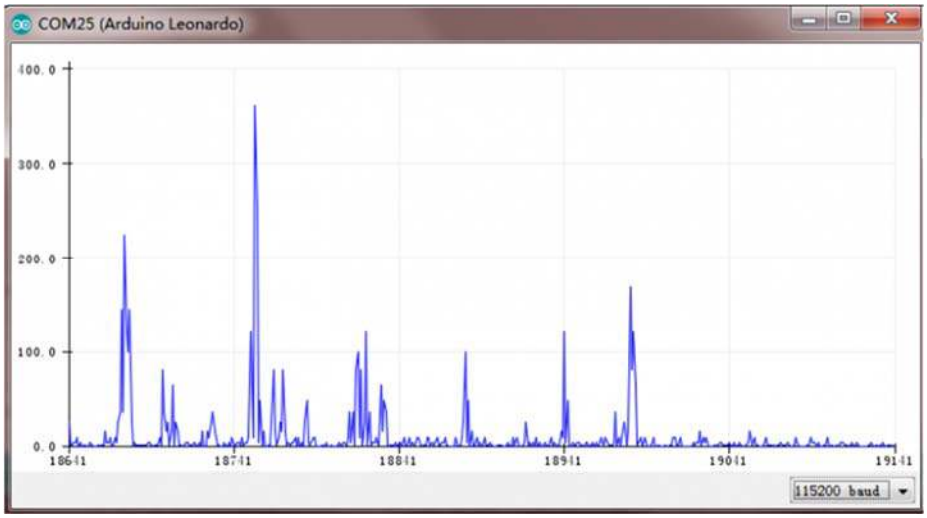
```
  The filters work only with fixed sample frequency of
  `SAMPLE_FREQ_500HZ` or `SAMPLE_FREQ_1000HZ`.
  Inputs at other sample rates will bypass
*/
SAMPLE_FREQUENCY sampleRate = SAMPLE_FREQ_500HZ;
/*
  Set the frequency of power line hum to filter out.
  For countries with 60Hz power line, change to "NOTCH_FREQ_60HZ"
*/
NOTCH_FREQUENCY humFreq = NOTCH_FREQ_50HZ;


void setup()
{
  myFilter.init(sampleRate, humFreq, true, true, true);
  Serial.begin(115200);
}


void loop()
{
  int data = analogRead(SensorInputPin);
  int dataAfterFilter = myFilter.update(data);  // filter processing
  int envelope = sq(dataAfterFilter);   //Get envelope by squaring t
he input
  envelope = (envelope > threshold) ? envelope : 0;    // The data s
et below the base value is set to 0, indicating that it is in a rela
xed state


  /* if threshold=0,explain the status it is in the calibration proc
ess,the code bollow not run.
     if get EMG singal,number++ and print
  */
  if (threshold > 0)
  {
    if (getEMGCount(envelope))
    {
      EMG_num++;
```

```
      Serial.print("EMG_num: ");
      Serial.println(EMG_num);
    }
  }
  else {
    Serial.println(envelope);
  }
  delayMicroseconds(500);
}


/*
   if get EMG signal,return 1;
*/
int getEMGCount(int gforce_envelope)
{
  static long integralData = 0;
  static long integralDataEve = 0;
  static bool remainFlag = false;
  static unsigned long timeMillis = 0;
  static unsigned long timeBeginzero = 0;
  static long fistNum = 0;
  static int  TimeStandard = 200;
  /*
    The integral is processed to continuously add the signal value
    and compare the integral value of the previous sampling to deter
mine whether the signal is continuous
   */
  integralDataEve = integralData;
  integralData += gforce_envelope;
  /*
    If the integral is constant, and it doesn't equal 0, then the ti
me is recorded;
    If the value of the integral starts to change again, the remainf
lag is true, and the time record will be re-entered next time
  */
  if ((integralDataEve == integralData) && (integralDataEve != 0))
```

```
  {
    timeMillis = millis();
    if (remainFlag)
    {
      timeBeginzero = timeMillis;
      remainFlag = false;
      return 0;
    }
    /* If the integral value exceeds 200 ms, the integral value is c
lear 0,return that get EMG signal */
    if ((timeMillis - timeBeginzero) > TimeStandard)
    {
      integralDataEve = integralData = 0;
      return 1;
    }
    return 0;
  }
  else {
    remainFlag = true;
    return 0;
   }
}
```

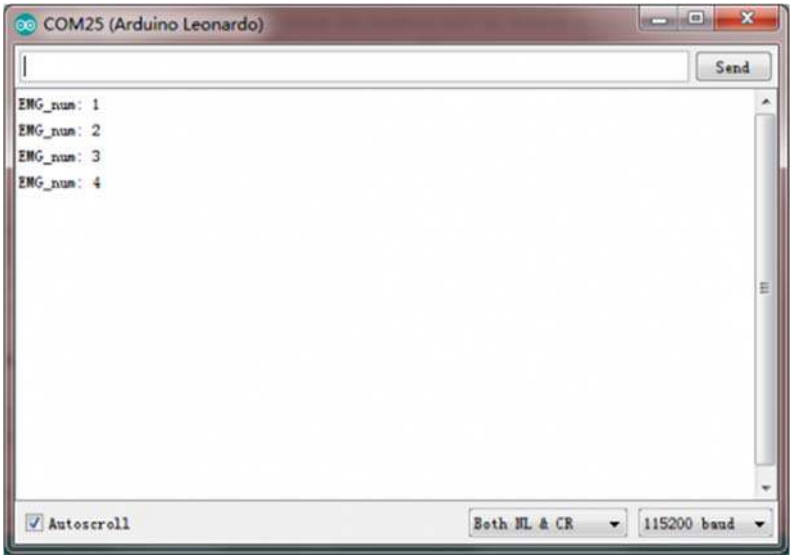
Calibration is recommended for every time you use it, for the EMG signal is different in different position even of the same person.

- 1. In this code, the default variable **threshold** is 0, which means in the calibration mode. Modify **threshold** to nonzero means exit the calibration mode. So if you want to re-calibrate the sensor,you just need to modify **threshold** value to 0, namely, **unsigned long threshold = 0;**
- 2. Upload the code to Arduino control board and open serial plotter of Arduino IDE to check the maximum Y-axis value.
- 3. Make a fist gently and loose it gently, repeatedly. Then observing and recording the maximum Y-axis value. As shown in the following picture,the result of the maximum should be around 400.



- 4. Modify the variable **threshold** to the recorded maximum value. Recommend to modify it to the maximum +100 to improve anti-interference. Reload the sample code to the Arduino control board.
- 5. Open the serial monitor of Arduino IDE, make a fist and loose it for once then you can see the printed value. Making a fist gently will not count and making a fist with more power will not increase the count value. Only a strong fist and loose can make sense. In this way, we realize the precise counts.

If you found the count accumulated when loose the fist, it would be caused by low threshold value. You can add extra 100 on the variable **threshold** , then reloading the sample code to the Arduino control board and check the result. You can repeat this step until the result is satisfied.



This sensor can count the muscle activities precisely after calibration on putting the sensor to a position. It can be applied to push-up counts, dumbbell curl counts. It also can be applied to human-computer interaction, such as play Flappy Bird with EMG signal made by fist.Besides the basic

function, the analog EMG sensor can also detect muscle activity strength by signal intensity. You can refers to the More Documents in the end page of Wiki.

FAQ

Q1. Where should I put the dry electrode? Is there any special requirement?

A. In general, the corresponding 3 metal dry electrode plate needs not pay attention to the reference voltage. It only need to consistent with the direction of muscle. You can refer to the image as below of the placement.

For any questions, advice or cool ideas to share, please visit the **DFRobot Forum**.