El repositorio [emg-processing-in-embedded-system] es un proyecto que tiene como objetivo procesar señales electromiográficas (EMG) en un sistema embebido basado en Arduino. El proyecto utiliza un sensor EMG de df-robots conectado a un pin analógico del Arduino, que envía los datos a un programa en Python que los grafica y los clasifica mediante un algoritmo de aprendizaje automático. El resultado de la clasificación se envía de vuelta al Arduino, que controla un servomotor según el valor procesado.

Para realizar este proyecto, se han apoyado en otros dos repositorios de procesamiento de señales EMG: [EMG-Signal-Processing-Library] y [emg_cvm_normalization]. Estos repositorios permiten aplicar diferentes técnicas de filtrado, normalización, extracción de características y reducción de dimensionalidad a las señales EMG, para mejorar su calidad y facilitar su clasificación.

El código de Arduino básico es::

```
#include <Servo.h>
float sensor;
unsigned long tiempoActualTemp1, tiempoAnteriorTemp1 = 0;
unsigned long tiempoActualTemp2, tiempoAnteriorTemp2 = 0;
int processedValue;
float s_lim=370;
float i_lim=220;

Servo servoMotor;

void setup() {
  Serial.begin(115200);
  servoMotor.attach(3);
}

void loop() {

  tiempoActualTemp1 = millis();
  tiempoActualTemp2= millis();

  if (tiempoActualTemp1 - tiempoAnteriorTemp1 >= 2) { //Tiempo en milisegundos;
    tarea1();
    tiempoAnteriorTemp1 = tiempoActualTemp1;
  }
  if (tiempoActualTemp2 - tiempoAnteriorTemp2 >= 1000) { //Tiempo en
milisegundos;
```

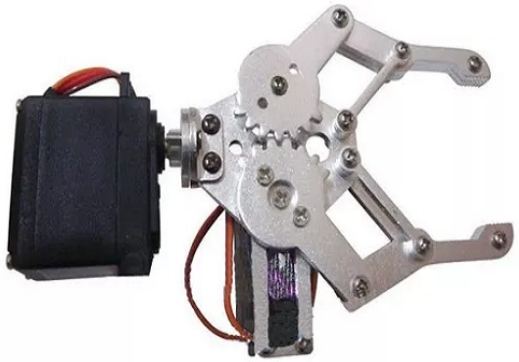
```
    tarea2();
    tiempoAnteriorTemp2 = tiempoActualTemp2;
  }
}

void tarea1() { // Envía datos a Python
  sensor = analogRead(A0);
  Serial.println(sensor);


  if (sensor > s_lim ||  i_lim > sensor){
    processedValue = 1;
  }
   else {  //if (sensor > 265 ||  335 > sensor)
    processedValue = 0;
  }
}

void tarea2() {

    if (processedValue == 0){
      servoMotor.write(50);
    }
    else{
      servoMotor.write(0);
    }

}
```

Este código realiza las siguientes funciones:

- Incluye la librería Servo.h para controlar el servomotor.
- Declara e inicializa las variables que se van a utilizar: sensor, tiempoActualTemp1, tiempoAnteriorTemp1, tiempoActualTemp2, tiempoAnteriorTemp2, processedValue, s_lim e i_lim.
- Crea un objeto servoMotor de la clase Servo y lo asocia al pin 3 del Arduino.
- En la función setup(), inicializa la comunicación serial a 115200 baudios y llama al método attach() del servoMotor para activarlo.
- En la función loop(), obtiene el tiempo actual en milisegundos y lo almacena en las variables tiempoActualTemp1 y tiempoActualTemp2.

- Si la diferencia entre tiempoActualTemp1 y tiempoAnteriorTemp1 es mayor o igual a 2 milisegundos, llama a la función tarea1() y actualiza el valor de tiempoAnteriorTemp1.
- Si la diferencia entre tiempoActualTemp2 y tiempoAnteriorTemp2 es mayor o igual a 1000 milisegundos, llama a la función tarea2() y actualiza el valor de tiempoAnteriorTemp2.
- La función tarea1() lee el valor del sensor EMG conectado al pin A0 y lo envía por el puerto serial. También compara el valor del sensor con los límites s_lim e i_lim, y asigna un valor de 1 o 0 a la variable processedValue según el resultado de la comparación.
- La función tarea2() escribe un valor de 50 o 0 en el servoMotor según el valor de processedValue, lo que hace que el servomotor gire o se detenga.

Para el accionar del servo se utilizó una pinza genérica para facilitar el trabajo

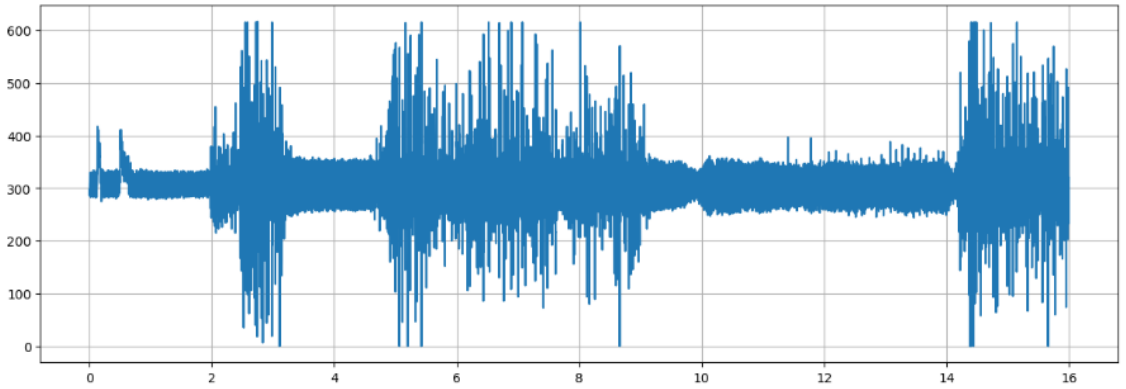


En la visualización de los datos se hace una conexión arduino-python para llegar a verlos en tiempo real o en un periodo determinado de tiempo

Para la calibración de los parámetros de uso de verlo en un tiempo acotado de tiempo para lograr visualizarlo mejor

In [3]:

```
tiempo = np.arange(0, seg, 0.002)  #Desde cero hasta X segundos en un muestreo de  500 muestras por segundo
#len(tiempo)

fm = 500 # 500[Hz] frecuencia de muestreo
T = 1/fm # Periodo del muestreo
largo_señal = len(l) # Largo de la señal 1
duracion_señal = T * largo_señal
tiempo_señal = np.arange(0, duracion_señal, T)

plt.figure(figsize = (15,5))

plt.subplot(111)
plt.plot(tiempo,l) #
plt.grid(True)
```



donde se puede ver fácilmente dónde se encuentra en reposo y en qué valores entra en excitación.