ECG capteur AD8232.

L'AD8232 est un capteur permettant de récupérer un signal analogique contenant un électrocardiogramme. Il permet **d'extraire le signal**, de **l'amplifier** ainsi que de le **filtrer.**

Voici le lien de sa datasheet :

https://www.analog.com/media/en/technical-documentation/data-sheets/ad8232.pdf

Ainsi que celui de sa schématique :

https://cdn.sparkfun.com/datasheets/Sensors/Biometric/AD8232_Heart_Rate_Monitor_v10.pdf?_gl=1*1kb93tx*_ga*MTcyNjQzNzc2My4xNjg1OTU1NTEw*_ga_T369JS7J9N*MTY5NDY4MjQ4Mi4xLjEuMTY5NDY4MjUxNy4yNS4wLjA.

Matériels nécessaires :

- Capteur AD8232
- Electrodes
- Microcontrôleur ESP32
- IDE Arduino (avec les librairies nécessaires pour l'utilisation de l'ESP32)
- Convertisseur DC-DC et une pile, pour l'alimentation de l'AD8232
- Connectiques (câbles usb pour l'ESP32 et câbles pour la connectique entre le capteur et l'ESP32)
- Point d'accès Wi-Fi

Objectifs du TP :

- Mise en place de la connectique du capteur et de l'ESP32
- Extraction et visualisation des informations du capteur
- Connection de l'ESP32 au Wi-Fi
- Envoi et visualisation des informations à distance

Petit point informatif avant de commencer :

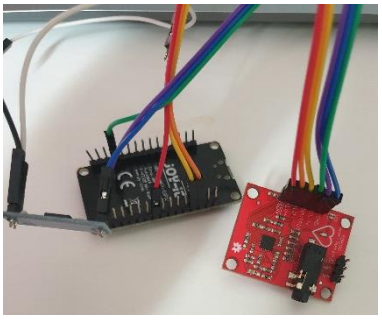
Le capteur AD8232 est très sensible à son alimentation. En effet, ce dernier capte des tensions qui sont très faible (de l'ordre du mV) l'alimentation peut très rapidement y injecter du bruit si elle n'est pas stable ou trop éloignée de la valeur nominale de l'alimentation du capteur. Ce qui résulte en un signal de sortie inexploitable car trop bruité.

C'est pour cela qu'une pile 9V ainsi qu'un convertisseur DC-DC sont utilisés pour l'alimentation de l'AD8232.

I – Connectique

Nous utiliserons sur le capteur AD8232 les broches suivantes :

- GND à connecter sur le GND du convertisseur nécessaire pour l'alimentation du capteur
- 3.3V à connecter sur le VOUT du convertisseur nécessaire pour l'alimentation du capteur
- OUTPUT à connecter sur la broche D34 de l'ESP32 (ou une autre broche disposant d'un ADC à son entrée) afin de récupérer le signal contenant l'ECG
- LO- et LO+ à connecter sur les broches D2 et D15 de l'ESP32 cela permet de détecter une erreur sur le branchement ou le positionnement des électrodes (voir datasheet)
- /SDN à connecter sur la broche D18 de l'ESP32 cela permet de sortir le capteur du mode économie d'énergie (dans lequel il ne fonctionne pas toujours bien)



Attention :

Bien sûr les broches indiquer ci-dessus sont à titre indicatif (ce sont celles utilisées dans le programme fourni). Vous pouvez utiliser celles que vous voulez, à condition de bien les initialiser dans le programme et de vérifier sur la datasheet de l'ESP32 qu'elles ont bien les caractéristiques dont vous avez besoin ! (par exemple un ADC pour la broche OUTPUT du capteur)

Il est également possible que certaines broches ne fonctionnent plus une fois l'initialisation du Wi-Fi faite, cela signifie qu'elles sont réservées pour l'utilisation du Wi-Fi (se référer à la datasheet), il vous reste plus qu'à changer de broche :-)

II – Extraction et visualisation des données

Une fois tous les branchements faits, il vous reste plus qu'à brancher l'ESP32 sur votre PC et lancer l'IDE Arduino.

Si les librairies nécessaires pour l'utilisation d l'ESP32 ne sont pas installées, il faut les installer.

Téléverser le code suivant sur l'ESP32 :

```
void setup() {
  // initialize the serial communication:
  Serial.begin(9600);
  pinMode(15, INPUT); // Setup for leads off detection LO +
  pinMode(2, INPUT); // Setup for leads off detection LO -
  pinMode(18 ,OUTPUT);
  digitalWrite(18, HIGH);
```

```
}

void loop() {
  if((digitalRead(15) == 1)||(digitalRead(2) == 1)){
    Serial.println('!');
  }
  else{
    // send the value of analog input 0:
      Serial.println(analogRead(34));
  }
  //Wait for a bit to keep serial data from saturating
  delay(1);
}
```

Ce code permet d'extraire et de visualiser en temps réel l'ECG. Pour cela vous pouvez utiliser l'outil « serial plotter » de l'IDE Arduino en le réglant sur 9600 bauds.

III – Connexion de la carte en Wi-Fi, envoi et visualisation des informations à distance

Avant de passer à cette étape, assurez-vous d'avoir toutes les librairies nécessaires pour la connexion au Wi-Fi de l'ESP32.

Téléverser le code suivant dans l'ESP32 :

```
#include <WiFi.h>
#include <NTPClient.h>
#include <ArduinoJson.h>
#include <HTTPClient.h>
#include <time.h>

#define wifi_ssid "rentrerNomWIFI"
#define wifi_password "rentrermdpWIFI"

const long gmtOffset_sec = 0;
const int daylightOffset_sec = 3600;

String idSensor = "ESP_0011"; //voir documentation de l'application
String idMeasurementType = "ecg"; /idem

long lastMsg = 0;
WiFiClient espClient;
const char* ntpServer = "pool.ntp.org";

  int nb=0;

void setup() {
  Serial.begin(115200);
  setup_wifi();
  configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
  pinMode(15, INPUT); // Setup for leads off detection LO +
  pinMode(2, INPUT); // Setup for leads off detection LO -
  pinMode(18 ,OUTPUT); //Pin pour passage hors du mode éco énergie
  digitalWrite(18, HIGH); // écriture à 1 pour cela
}

void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connexion à ");
```

```
  Serial.println(wifi_ssid);
  WiFi.begin(wifi_ssid, wifi_password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("Connexion WiFi établie");
  Serial.print("=> Adresse IP : ");
  Serial.println(WiFi.localIP());
}

void httpPOST() {


  struct tm timeinfo;
  if(!getLocalTime(&timeinfo)){
    Serial.println("Failed to obtain time");
    return;
  }

  struct tm *timestamp=&timeinfo;
  char buffer [25];
  strftime(buffer, 25, "%FT%T.000Z", timestamp);
  //Serial.println(buffer);


  if (WiFi.status() == WL_CONNECTED) {
    WiFiClient client;
    HTTPClient http;

    http.begin("https://dev-api.rami.ig.umons.ac.be/api/v1/measurements/bulk");

    http.addHeader("Content-Type", "application/json");


// Taille maximale du document JSON
  // Ajustez cette valeur en fonction de vos besoins
  int size = 200;
  const size_t capacity = JSON_ARRAY_SIZE(size) + size*JSON_OBJECT_SIZE(4);

  // Créer un document JSON
  DynamicJsonDocument documentTableau(capacity);

  // Créer un tableau JSON
  JsonArray Mesures = documentTableau.createNestedArray();

  // Ajouter des éléments au tableau
  StaticJsonDocument<110> documentUneMesure;
  for (int i = 0; i < size-1; i++) {
    documentUneMesure["value"] = analogRead(34);
    documentUneMesure["date"]= buffer;
    documentUneMesure["type"]= idMeasurementType;
    documentUneMesure["sensor"]= idSensor;
    String uneMesure;
    //serializeJson(documentUneMesure, uneMesure);
    Mesures.add(documentUneMesure);
    delay(1);
  }

  // Convertir le document JSON en chaîne JSON
    String leTableau;
    serializeJson(Mesures, leTableau);
    int httpResponseCode = http.POST(leTableau);
```

```
    Serial.print("HTTP Response code: ");
    Serial.println(httpResponseCode);
    http.end();
  }
  else {
    Serial.println("WiFi Disconnected");
  }
}

void loop() {

  httpPOST();
}
```

Ce code permet de se connecter au point Wi-Fi spécifié, puis de faire des post http contenant un fichier au format JSON contenant lui-même les valeurs de l'ECG.
Pour l'envoi et la visualisation de ces données du côté serveur, voir la documentation de l'application qui a été faite.