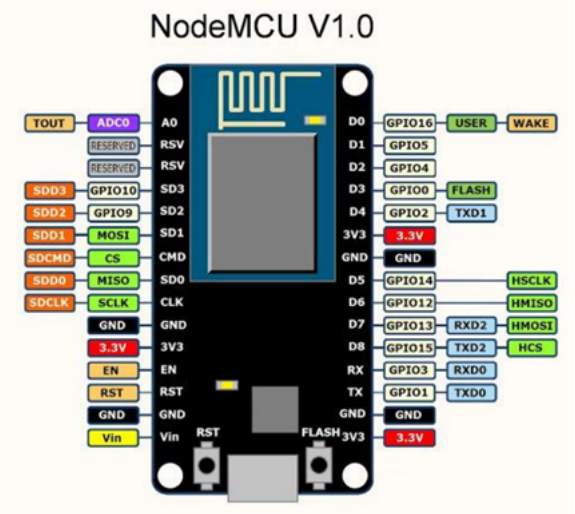
Rapports de projets d'EMBI4 n°1-2

Mora Leonardo, Sabouri Abdessamad

Dans ce cours introduit par notre professeur M. Georges, nous allons découvrir et expérimenter pendant les laboratoires, les micro-contrôleurs ainsi que leurs utilisations. Nous utiliserons l'ESP-12F comportant la puce ESP8266.



Ce micro-contrôleur fonctionne comme un arduino simple. il est connstitué d'un module WiFi, du bluetooth, possèdent des pins PWM, I2C,3.3v (uniquement), GND, pins configurable IN/OUT, ...

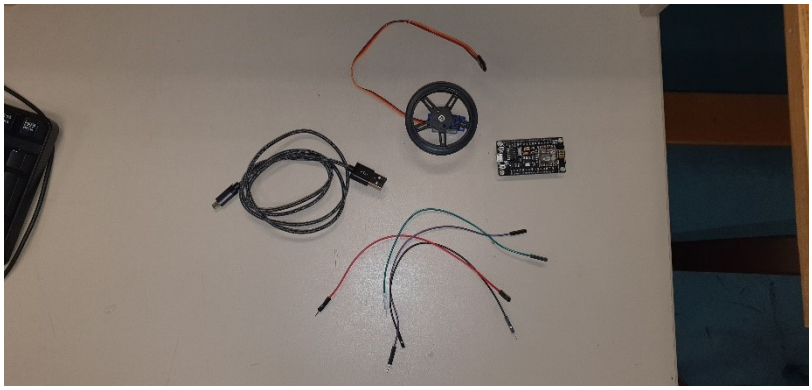
1 Utilisation d'un servo moteur

Le 1^{er} projet consiste à l'utilisation et à la compréhension d'un servo moteur. Il existe plusieurs catégories de servo moteurs, mais dans notre cas nous utiliseront un servo moteur ne pouvant pas être dirigée, mais à se mouvoir en ligne droite. Nous créeront un code permettant de faire tourner la roue attachée au servo moteur. Ensuite nous utiliseront un oscillateur afin de d'observer les différents comportements selon les paramètres introduits.

1.1 Matériel:

il comporte:

- un servo moteur avec une roue attachée
- un ESP-12F
- un câble d'alimentation



1.2 Assemblage:

Le servo moteur comprend 3pins:

- le pin VIN, servant à l'alimentation du servo moteur
- le pin GND pour la "terre"
- le pin pour une entrée digitale

On connectera le le pin VIN au pin 3.3V de l'ESP, le pin GND au GND de l'ESP et le pin d'entrée à un des pins de l'ESP en n'oubliant pas de configurer le pin en sortie de l'ESP, dans notre cas on le connectera à D2 de l'ESP.(Et connecter câble d'alimentation à l'ESP).

1.3 Code Arduino:

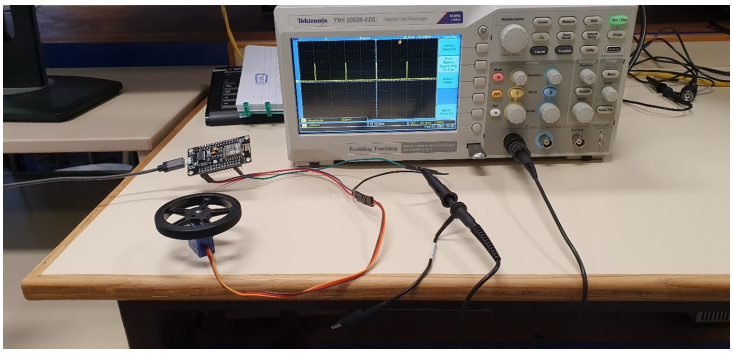
```
#include <Servo.h>

Servo myServo;
```

```
void setup(){
    Serial.begin(9600);
    myServo.attach(4);
}

void loop(){
    myServo.write(179);
    delay(1500);
    myServo.write(01);
    delay(1500);
}
```

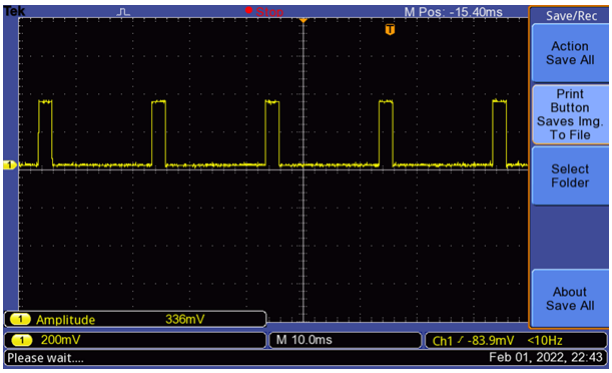
Nous utilisons la librairie Servo.h pour les servo moteurs dans ce code, nous envoyons un paramètre, puis un paramètre opposé par intervalle d'1.5s.

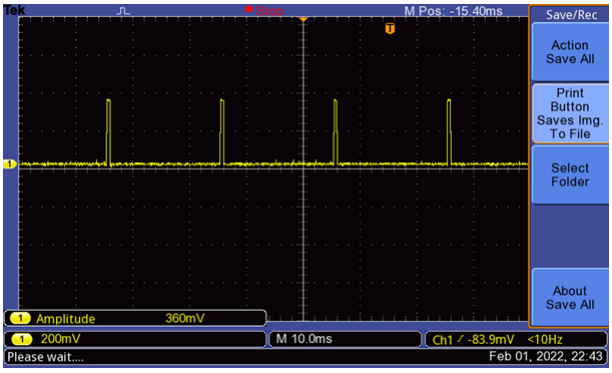
1.4 Observation et explication:

L'utilisation de 2 paramètres distincts permets d'observer clairement la différence de vitesse. Ce servo moteur tourne en fonction du paramètre entrée. Si le composant se trouve entre 1 et 179, alors elle tournera d'un coté, et de 180 à 359,de l'autre côté. Dans un sens plus le paramètre sera élevé, plus la roue tournera vite. Cela s'explique car le paramètre défini la fréquence envoyé au servo moteur. On peut le distinguer grâce à l'oscilloscope.

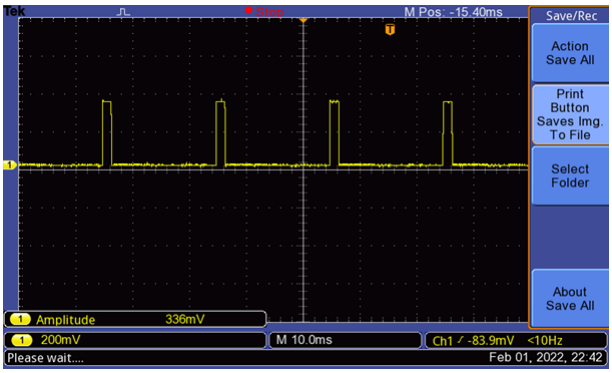


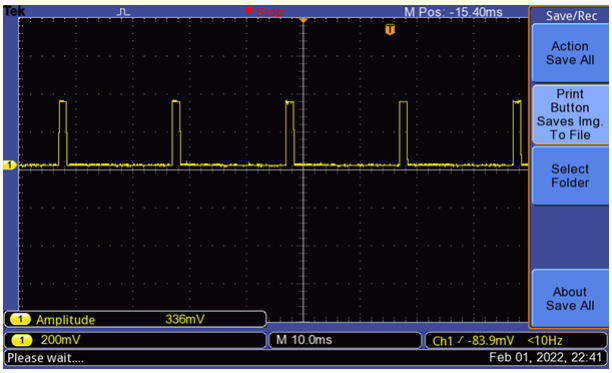
Dans les $2^{\grave{e}^{res}}$ images, On fait varier le paramètre entre 1 et 179, on distingue parfaitement la différence entre les deux et le changement fait entre les deux après l'intervalle de temps.





Les 2 dernières images correspondent à des paramètres de 100 et 80 à intervalles régulier. On remarque que la fréquence de signal est très rapprochée, ce qui est assez difficile à les différencier, et leurs vitesses sont assez proche.





2 Utilisation d'un capteur de température DS18B20

Dans ce $2^{ème}$ projet, nous allons expérimenter l'utilisation du capteur de température DS18B20 avec l'ESP-12F dans un premier temps. Ensuite nous allons créer un serveur https ayant son propre réseau WiFi et passer le résultat de température dans la page créée.

2.1 Le Capteur DS18B20:

Le capteur DS18B20 est un capteur de température numérique à un fil, cela signifie qu'il ne nécessite qu'une seule ligne de données (et de GND) pour communiquer avec l'ESP. Il peut être alimenté par une alimentation externe ou être alimenté par la ligne de données (appelé "mode parasite"), ce qui élimine du besoin d'une alimentation externe.

2.2 Matériel:

Il comporte:

- un capteur DS18B20
- un ESP-12F
- des cables M/F
- un breadboard
- une résistance

2.3 Assemblage:

On connecte le GND du capteur à celui de l'ESP, le pin d'alimentation au 3.3v de l'ESP et le pin de communication avec un des pins digital La résistance fera le pont entre le le pin d'alimentation et le pin de communication. Un breadboard nous facilitera en créant un assemblage propre.

2.4 Code Arduino:

```
    #include <OneWire.h>


/* Broche du bus 1-Wire */
const byte BROCHE_ONEWIRE = 4;

/* Code de retour de la fonction getTemperature() */
enum DS18B20_RCODES {
  READ_OK,  // Lecture ok
  NO_SENSOR_FOUND,  // Pas de capteur
  INVALID_ADDRESS,  // Adresse reçue invalide
  INVALID_SENSOR  // Capteur invalide (pas un DS18B20)
};


/* Création de l'objet OneWire pour manipuler le bus 1-Wire */
OneWire ds(BROCHE_ONEWIRE);


/**
 * Fonction de lecture de la température via un capteur DS18B20.
 */
byte getTemperature(float *temperature, byte reset_search) {
  byte data[9], addr[8];
  // data[] : Données lues depuis le scratchpad
  // addr[] : Adresse du module 1-Wire détecté

  /* Reset le bus 1-Wire ci nécessaire (requis pour la lecture du premier capteur) */
  if (reset_search) {
    ds.reset_search();
  }

  /* Recherche le prochain capteur 1-Wire disponible */
  if (!ds.search(addr)) {
    // Pas de capteur
    return NO_SENSOR_FOUND;
  }
```

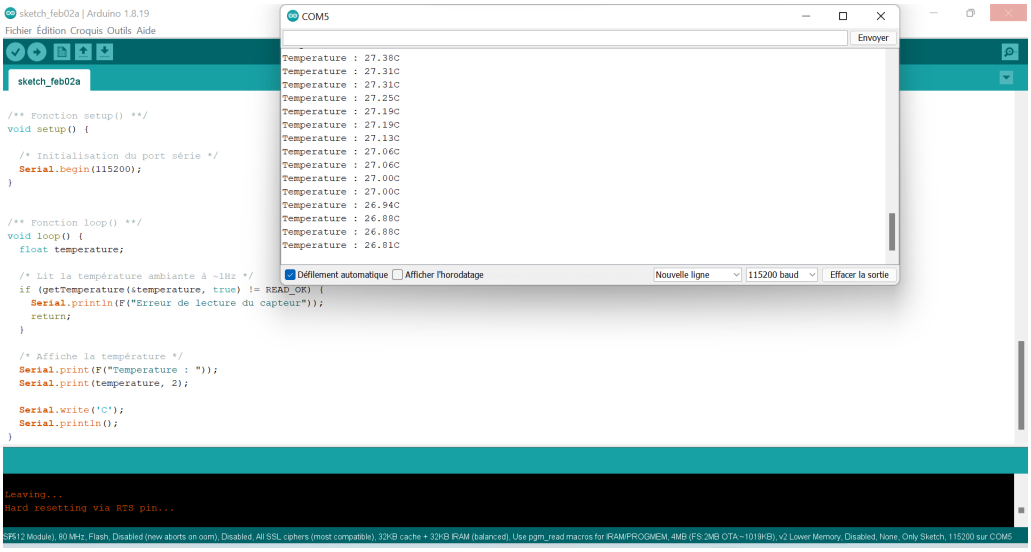
```
  /* Vérifie que l'adresse a été correctement reçue */
  if (OneWire::crc8(addr, 7) != addr[7]) {
    // Adresse invalide
    return INVALID_ADDRESS;
  }


  /* Reset le bus 1-Wire et sélectionne le capteur */
  ds.reset();
  ds.select(addr);

  /* Lance une prise de mesure de température et attend la fin de la mesure */
  ds.write(0x44, 1);
  delay(800);

  /* Reset le bus 1-Wire, sélectionne le capteur et envoie une demande de lecture du scratch
  ds.reset();
  ds.select(addr);
  ds.write(0xBE);

 /* Lecture du scratchpad */
  for (byte i = 0; i < 9; i++) {
    data[i] = ds.read();
  }

  /* Calcul de la température en degré Celsius */
  *temperature = (int16_t) ((data[1] << 8) | data[0]) * 0.0625;

  // Pas d'erreur
  return READ_OK;
}


/** Fonction setup() **/
void setup() {

  /* Initialisation du port série */
  Serial.begin(115200);
}


/** Fonction loop() **/
void loop() {
  float temperature;
```

```
  /* Lit la température ambiante à ~1Hz */
  if (getTemperature(&temperature, true) != READ_OK) {
    Serial.println(F("Erreur de lecture du capteur"));
    return;
  }

  /* Affiche la température */
  Serial.print(F("Temperature : "));
  Serial.print(temperature, 2);

  Serial.write('C');
  Serial.println();
}
```

2.5 Observation et explication:

Le code est composé de deux parties:

- La $1^{ère}$ partie est la température. Nous devons convertir ce que nous recevons en degré, et affiche en boucle la température (il update). Lors du test nous recevions 28°c ce qui constitue la température de la pièce.



- La $2^{ème}$ partie est la mise en place d'un serveur web. Nous initialisons au départ le port 80 du serveur. Nous créons avec un spot WiFi dont nous nous connecterons à une adresse ip pour pouvoir afficher sur un site html, la température courante. Pour faire la mise à jour, nous devons rafraîchir la page.