Practica 1: protocolos

Tareas:
1. Leer repetidamente el valor del potenciómetro obtenido por el Arduino.
2. Leer repetidamente el valor analógico del potenciómetro e indicar el ángulo de posicionamiento de la perilla.
3. Leer repetidamente el valor analogico del potenciometro e indicar el ángulo de posicionamiento de la perilla, a una frecuencia de 100 Hz

```
int analogPin = 0;      // potentiometer wiper (middle terminal) connected to
    analog pin 3
                        // outside leads to ground and +5V
int analog_val = 0;     // variable to store the value read

const float step_deg = 0.286;
long int T_total = 10000;

void setup()
{
  Serial.begin(115200);               //  setup serial
}

void loop()
{
  long int t1, t2;
  t1 = micros();
  float deg = 0;
  analog_val = analogRead(analogPin);  // read the input pin
  deg = step_deg * analog_val;

  Serial.println(deg);
  t2 = micros();
  //Serial.print("Time taken by the task: "); Serial.print(t2-t1);
    Serial.println(" milliseconds");
  delayMicroseconds(T_total - (t1-t2));      // debug value
}
```

1. Comandar una señal de PWM al servo.
2. Comandar una señal de ángulo al servo.
3. Comandar una señal de ángulo al servo, utilizando como referencia el ángulo del potenciómetro, a una frecuencia de 100 Hz.

4. Comandar una señal de ángulo al servo, utilizando como referencia el ángulo del potenciómetro, a frecuencias de 100 Hz, 50 Hz, 10 Hz y 1 Hz y analizar las diferencias

```
#define MICROS2SEC 1000000
#define MILIS2SEC  1000
#define SERVO 9
int pote = A0;

int grados = 90;              //angulo incial servo
int angleMin = 0;             //angulo minimo servo
int angleMax = 180;           //angulo maximo servo

//unsigned int duty_t = 1000;  // 1ms=1000us(valor inicial)
const long dutyMin   = 600;    // 1ms=1000us
const long dutyMax   = 2600;   // 2ms=2000us
const long servoFreq = 20000;  // 20ms=20000us -> 50Hz

//Para el ultimo item.
const int frecuencias[] = {100, 50, 10, 1};
const int len_hz = sizeof(frecuencias) / sizeof(frecuencias[0]);


//1. Comandar una señal de PWM al servo.
void comandarPWM() ;

//2. Comandar una señal de ángulo al servo.
void comandarAngulo(int grados, int angle_min, int angle_max) ;

//3. Comandar una señal de ángulo al servo, utilizando como referencia el ángulo
//   del potenciómetro, a una frecuencia de 100 Hz.
void comandaAnguloFromPote(int frecuencia, int angle_min, int angle_max);

void setup() {
  Serial.begin(115200);
  pinMode(SERVO, OUTPUT);

  // Configuración del TIMER1
  TCCR1A = 0;                // Limpiar el registro de control A
  TCCR1B = 0;                // Limpiar el registro de control B
  TCNT1 = 0;                 // Inicializar el temporizador
```

```
  OCR1A = 1249;               // Valor para generar una frecuencia de 50 Hz: 16Mhz/(50*256) -1 = 1249
  TCCR1B |= (1 << WGM12);     // Modo CTC (Clear Timer on Compare Match)
  TCCR1B |= (1 << CS12) ;     // Prescaler de 64: CS12 = 1, CS11 = 0 y CS10 = 0
  TIMSK1 |= (1 << OCIE1A);    // Habilitar interrupción por coincidencia de comparación en OCR1A

}

void loop()
{
  //comandarPWM();

  comandarAngulo(90, 0, 180);

  /**
  float tick = micros();
  comandaAnguloFromPote(100, 0, 180); //100hz, 50hz, 10hz, 1hz
  float tock = micros();
  Serial.println(1000000/(tock-tick));
  **/

}


//1_
void comandarPWM()
{
  noInterrupts();
  grados = 90;
  angleMin = -90;
  angleMax = 90;
  interrupts();

}

//2_
void comandarAngulo(int angulo, int angle_min, int angle_max)
{
  noInterrupts();
  angleMin = angle_min;
  angleMax = angle_max;
  grados = constrain(angulo, angleMin, angleMax);
```

```
  interrupts();
}

//3_
void comandaAnguloFromPote(int frecuencia, int angle_min, int angle_max)
{
  int minPote = 0;
  int maxPote = 292;    //290°-300°
  //int angle_min = -90; //min angulo del servo
  //int angle_max = 90;  //maximo angulo del servo
  unsigned long t0, t1;

  t0 = millis();
  //Obtengo el angulo con el pote.
  int angulo = map(analogRead(pote), 0, 1023, minPote, maxPote); // Lee el
ángulo del potenciómetro
  int grados_servo = map(angulo, minPote, maxPote, angle_min, angle_max);

  //Envio la informacion al servo.
  comandarAngulo(grados_servo, angle_min, angle_max);

  //Aseguro la frecuencia de deseada
  t1 = millis();
  delay((MILIS2SEC / frecuencia) - t1 + t0); //1hz->1000000us
}



ISR(TIMER1_COMPA_vect) {
  unsigned int t0, t1;
  t0 = micros();
  unsigned long duty_t = map(grados, angleMin, angleMax, dutyMin, dutyMax);

  // Generar pulso PWM
  digitalWrite(SERVO, HIGH);
  delayMicroseconds(duty_t);
  digitalWrite(SERVO, LOW);
  t1 = micros();
  delayMicroseconds(servoFreq -t1 + t0 ); // Tiempo restante del periodo (20 ms
- ancho del pulso)
}
```

1. ¿Cuál es el error de medición del ángulo del potenciómetro?

El error de medición del ángulo del potenciómetro viene dado por varios factores:
No se conoce con precisión el ángulo máximo del potenciómetro, este en si no llega a 360° sino que que oscila (dependiendo el potenciómetro) entre los 290° a 300°, en nuestro caso se midió un ángulo máximo aproximado de 292°.
Además, se debe tener en cuenta que se mide mediante una entrada analogica con 10 bits de precision en donde si se omite el ruido de medición (usualmente reduce a 8 bits efectivos) se puede medir ángulos cada $292°1024 = 0.286°$.

2. ¿Cuál es el error de posición comandada del servo?

El error de posición comandada del servo tiene que ver con la conversión de grados a tiempo del duty cicle, este duty va de 0.6ms a 2.6ms osea 2 ms en los cuales tiene que mapear los 180° que puede recorrer el servo.

asumiendo que el tiempo de muestreo no es un problema el error es $180°2000\ \mu s = 0.09[°/\mu s]$ pero además se debe tener en cuenta el “Dead band width: 10 µs” que tiene el servo SG90 usado, esto quiere tengo +/- 5 µs de error alrededor del pulso por lo que si tengo un pulso de 1500µs entonces el servo no se moverá si el pulso varia entre 1495 µs a 1505 µs, en grados sería 0.45°.

3. Cual es el error de posición del servo comandado desde el potenciómetro?

El error del servo comandado por el potenciómetro viene dado por una acumulacion de los puntos anteriores en donde con los razonamientos más optimistas se tiene un error de $0.45° + 0.286° = 0.76°$ mapeando los 292° a 20ms del duty cycle.