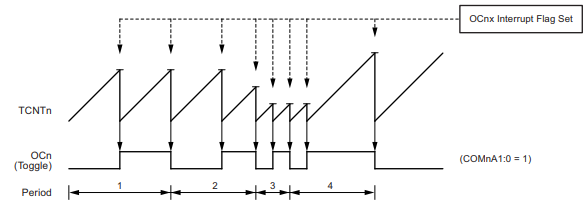
Control del ángulo de disparo

Para el control del ángulo de disparo se implementó el microcontrolador Arduino uno,

La primera parte consiste en la detección del cruce por cero, a partir del circuito xx, este envía un pulso cada vez que se detecta un cruce por cero, como se trata de una señal senoidal, se detecta un cruce por cero cada 180 grados, esta señal es principalmente usada para que el microcontrolador sepa cuando comienza un nuevo periodo y para que se sincronice con la señal de entrada. El cruce por cero activa en el microcontrolador una función ISR para que el contador TCNT1 empiece su conteo de nuevo, lea a que ángulo se requiere disparar y todo el proceso comience de nuevo en el siguiente periodo.

A partir de la primera lectura del cruce por cero, el microcontrolador comienza a contabilizar el tiempo transcurrido desde el Timer 1, a través del registro TCNT1. Cuando el tiempo es equivalente al ángulo donde se requiere realizar un disparo el registro TCNT1 se compara con el registro OCR1A que es el almacena el valor al que se va realizar la comparación, cuando esto ocurre se activa la interrupción por timer que inmediatamente se dirige a un espacio de código donde se procede a ejecutar el disparo de la onda R, al tiempo el registro TCNT1 comienza su conteo nuevamente, seguidamente se actualiza el valor de OCR1A para que contabilice cuanto tiempo permanecerá el disparo R en alto (20us), después se procede a igualar el registro OCR1A con el tiempo equivalente a 120 grados, para realizar el disparo de la onda S y el mismo proceso para el disparo de la onda T. esto es posible gracias a que el desfase de estas señales siempre es de 120 grados.

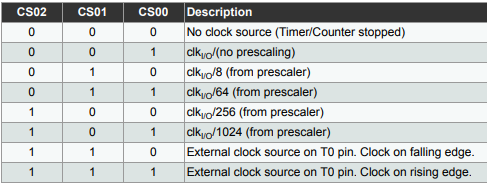
La siguiente imagen explica visualmente que sucede con el registro TCNT1 cuando se compara con el registro OCR1A.



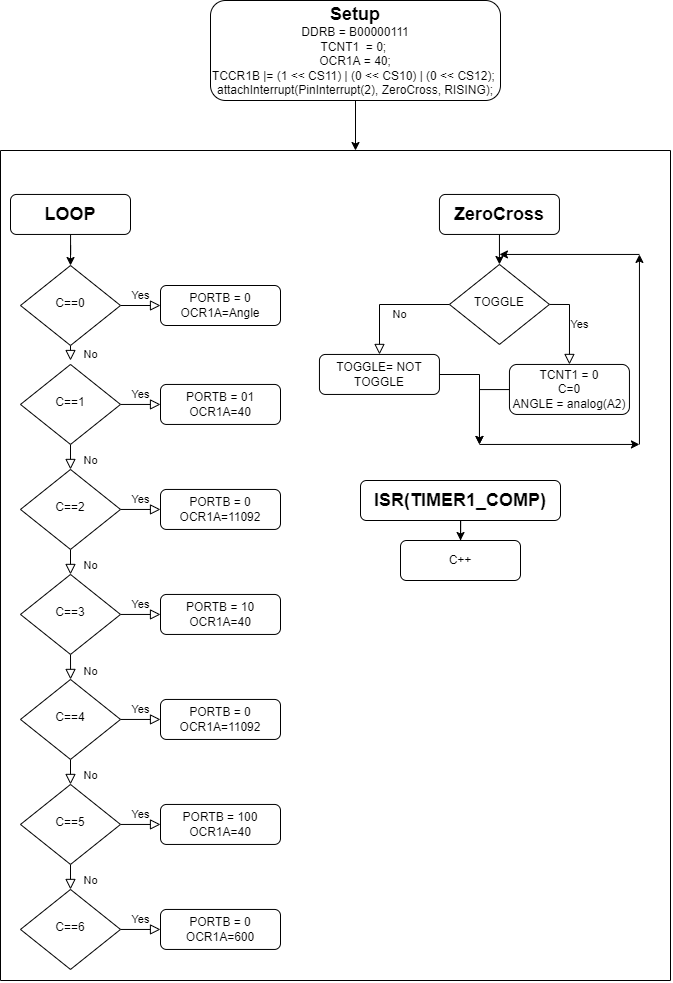
Para calcular la cantidad de veces que tiene que contar el registro, se tienen que tener en cuenta que:

La frecuencia de reloj es de 16MHz

El Timer 1 es de 16 bits, es decir que puede contar hasta

Contamos con los prescalers : 1024 – 256 – 64 – 8 cómo se puede observar en la siguiente tabla. 

La selección del prescaler es a partir de la cantidad de tiempo mínimo y máximo que se requiere contabilizar entre mayor sea el prescaler mayor tiempo podrá contabilizar pero con una sensibilidad menor y entre menor sea el prescaler menor tiempo podrá contabilizar pero su sensibilidad será mucho mayor, por esta razón se selecciona el prescaler de 8.



//Juan Camilo Serrano Correa

int c = 0;

bool Toggle = false;

int Angle = 8333;//2500\*x/27

int OverAngle = 0;

float Courrent = 0;

float Courrent1 = 0;

void setup()

{

Serial.begin(9600);

Serial.println(Angle);

Serial.print("");

DDRB = B00100111;//PORTB D8-D10 OUTPUT

// Configuracion de TIMER1

TCCR1A = 0; // limpia registro

TCCR1B = 0; //limpia registro

TCNT1 = 0; //Inicializa el temporizador --------------

OCR1A = 4000; // carga el registrador de comparación: (16MHz/1024\*1Hz) -1 = 15624 = 0X3D08

TCCR1B |= (1 << WGM12) | (1 << CS11) | (0 << CS10) | (0 << CS12); // modo CTC, prescaler de 8: CS12 = 1 e CS10 = 1

TIMSK1 |= (1 << OCIE1A); // habilita interrupción por igualdade de comparación

attachInterrupt(digitalPinToInterrupt(2), ZeroCross, RISING);

}

void loop()

{

if (Serial.available() != 0) {

//Angle = Serial.parseInt();

Angle = Serial.parseInt();

Serial.println(Angle);

}

if (Angle <= 11000) {

if (c == 0) {

PORTB = B00000000;

OCR1A = Angle;

//Angle=map(analogRead(A2), 0, 1023, 2778, 13886);

} else if (c == 1) {

PORTB = B00100001; //R

OCR1A = 40;

} else if (c == 2) {

PORTB = B00000000;

OCR1A = 11052;

} else if (c == 3) {

PORTB = B00100010; //S

OCR1A = 40;

} else if (c == 4) {

PORTB = B00000000;

OCR1A = 11052;//

} else if (c == 5) {

PORTB = B00100100; //T

OCR1A = 40;

} else if (c == 6) {

PORTB = B00000000;

//Courrent = analogRead(A0);

//Courrent1 = analogRead(A1);

OCR1A = 600;

}

} else {

OverAngle = Angle - 11000;

if (c == 0) {

PORTB = B00000000;

OCR1A = OverAngle;

//Angle=map(analogRead(A2), 0, 1023, 2778, 13886);

} else if (c == 1) {

PORTB = B00100100; //T

OCR1A = 40;

} else if (c == 2) {

PORTB = B00000000;

OCR1A = 11052;

} else if (c == 3) {

PORTB = B00100001; //R

OCR1A = 40;

} else if (c == 4) {

PORTB = B00000000;

OCR1A = 11052;//

} else if (c == 5) {

PORTB = B00100010; //S

OCR1A = 40;

} else if (c == 6) {

PORTB = B00000000;

//Courrent = analogRead(A0);

//Courrent1 = analogRead(A1);

OCR1A = 600;

}

}

}

ISR(TIMER1\_COMPA\_vect) // interrupción por igualdade de comparación en TIMER1

{

c++;//Address

}

void ZeroCross() {

if (Toggle) {//Reset when 360 cero crossing

TCNT1 = 0;//Reset timer

c = 0; //Reset counter

}

Toggle = not Toggle;//To ignore 180 cero crossing

}

// Angle = map(analogRead(A2), 0, 1023, 2778, 13886);//30-150

//PORTB = B00000001; //63ns

//digitalWrite(R, HIGH); //4us

https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P\_Datasheet.pdf