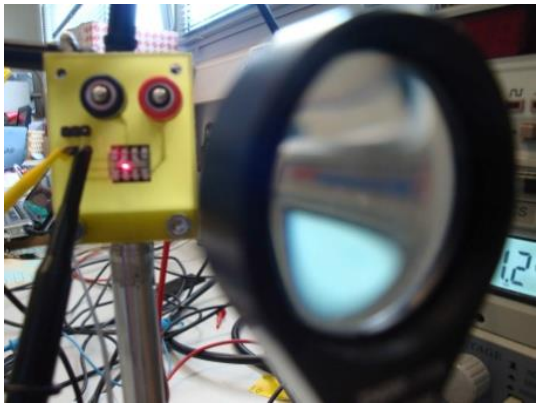
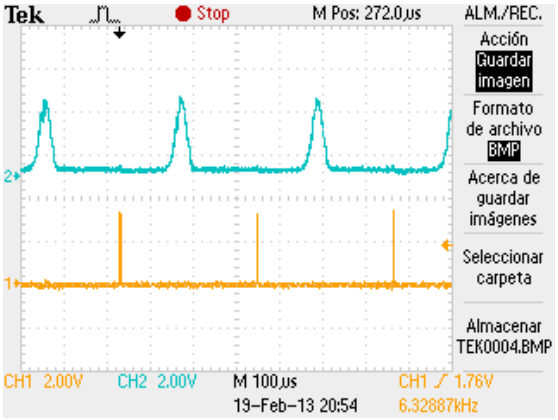
LABORATORI

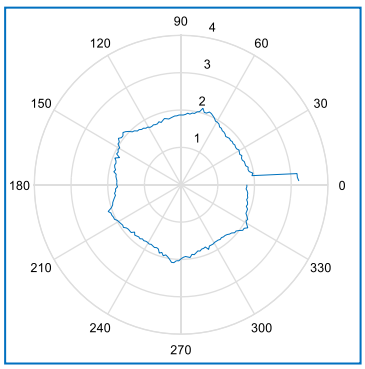
Laboratori Sistemes de Suport a la Informació.
SSTIC (Sistemes Suport per les TIC)

Miniproyecto

Desarrollo de una aplicación de Medida de distancias por Triangulación Óptica. Obtención de la Sección de un Objeto en Rotación







Por: Mauricio Moreno Sereno

Agradecimientos:

Francisco Palacio Bonet
Iván Bernat Ubiaga
Fernando Arreza Martin

Índice

Bibliografía:

http://taosinc.com for TAOS products (= http://ams.com)http://www.microchip.com

http://www.ti.com/product/MSP430F5529

Objetivos:

1. Conocer la técnica de triangulación óptica.
2. Conocer el array de fotodetectores TAOS TSL1401.
3. Conocer el entorno de programación MPLAB.
4. Programar las señales de control y comunicación serie con MICROCHIP – Microstick
5. Realizar un montaje óptico susceptible de medir distancias.

1. Introducción.

Dentro de las técnicas ópticas para medida de distancias se tiene:

i) la medida por **tiempo de vuelo** (*radar laser*), para medias y largas distancias. Se mide el tiempo de retorno de un pulso de luz.
ii) la **triangulación óptica**, para cortas y medias distancias, que se describe más adelante.
iii) la **interferometría**, para medidas en el rango de milímetros y con precisiones de nanómetros, basada en las interferencia de un haz de luz de referencia y un haz de luz "reflejado" por el objeto a medir.

Los sistemas basados en ultrasonidos son de bajo coste pero presentan algunas desventajas: i) el haz emitido es muy ancho y presenta poca resolución y ii) se pueden producir reflexiones especulares.

2. La triangulación óptica.

La técnica de medida de distancias por triangulación óptica se representa en la figura 1. En la parte de transmisión se tiene un láser y la óptica apropiada para focalizar el láser. En el punto o_1, donde se produce el impacto del láser en el objeto, se produce la difusión de éste. La luz dispersada por el objeto que atraviesa la lente 2 se hace converger mediante una lente adecuada sobre el punto a_1 en el dispositivo PSD (Position Sensor Device). Si el objeto a medir se encuentra en la posición o_2, a una distancia D del caso anterior, el punto imagen en el PSD a través del sistema óptico es el punto a_2. Fijada la geometría del sistema, el rango de medida, o_1-o_2, vendrá determinado por el tamaño del dispositivo lineal de medida a través de las posiciones a_1-a_2.

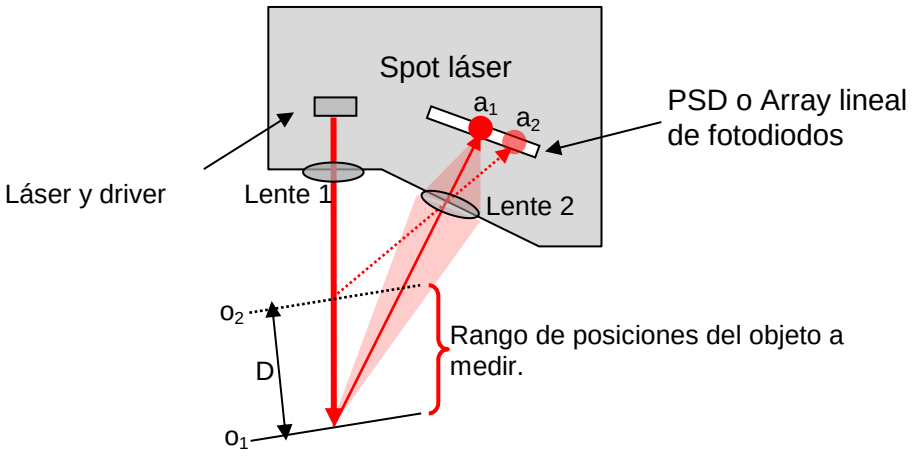


Figura 1. Esquema de un sistema por triangulación láser.

La distancia a medir se puede expresar en función de la posición del haz reflejado sobre el PSD o array de detectores. De forma muy aproximada:

$$D = \frac{f \cdot a}{x}$$

Siendo:

D	distancia a medir.
a	distancia entre emisor y receptor
f	focal de la lente en el receptor.
x	posición sobre el PSD en el que se focaliza el haz reflejado

Puede existir algún problema de linealidad entre la distancia al objeto y la posición en el dispositivo sensor. Para mantener un buen enfoque en todo el campo de profundidad del detector se debe satisfacer la condición de ***Scheimpflug***, que describe la orientación en el foco del sistema (cámara) ***cuando la lente no está paralela al plano imagen***.

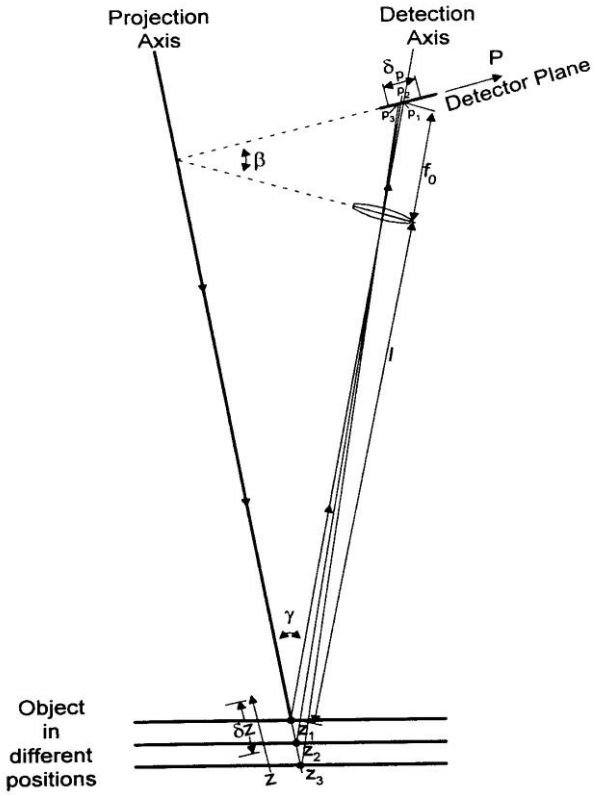
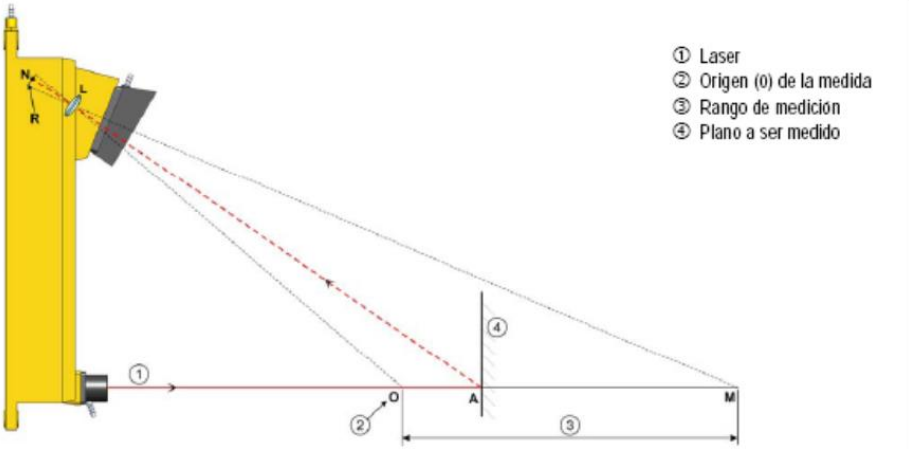


Figura 2. Condición de Scheimpflug.

2.1 Ejemplos comerciales.



Modelos / Rango de medición

Modelo		TL051 • TL021 • TL101 •	TL052 • TL022 • TL102 • TL112 •	TL053 • TL023 • TL103 •	TL057 • TL027 • TL107 • TL117 •	TL104 • TL114 •	TL105 •
Rango de medición	mm	500	1000	2000	3000	5000	8000
Distancias a partir del plano de fijación del sensor	mm	700 - 1200	1200 - 2200	700 - 2700	1700 - 4700	2000 - 7000	2000-10000
Precisión	± mm	0.4	1	3	4 (TL 05 : 7)	7	15
Tiempo de respuesta	ms	5 ms					

A pedido se ofrecen otras distancias y precisiones. (TL • • #• • soluciones)

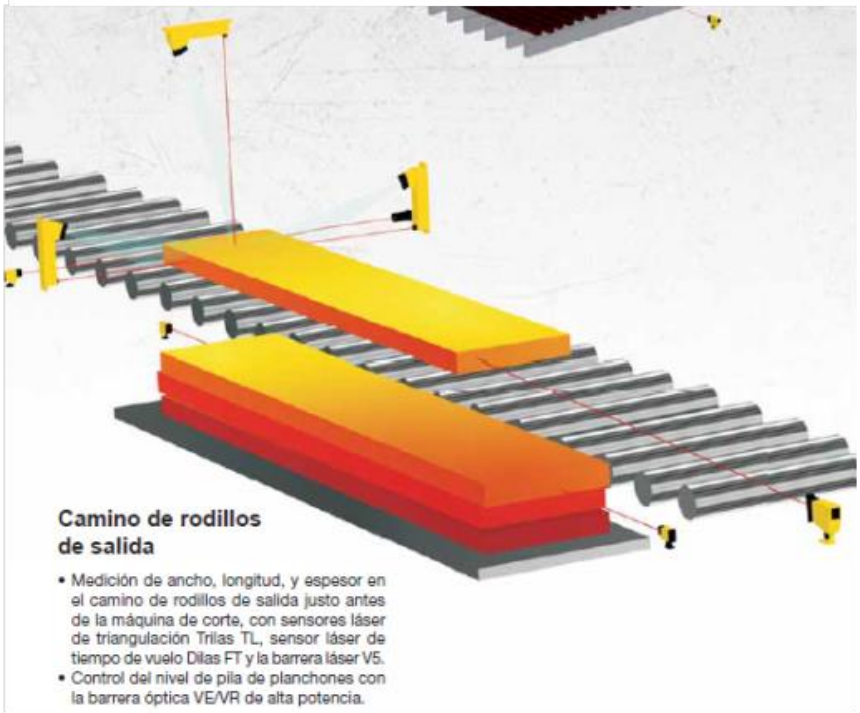


Figura 3. DELTA SENSORS™.
Sensores y Sistemas para la Industria Siderúrgica

Series LG5/LG10 Series Q50
The design of the sensor series LG5/LG10 and Q50 operates on **optical triangulation**. The Q50 uses an LED as a light source, whereas the LG5 and LG10 series consists of laser sensors. The emitter circuitry and optics create a light beam which is directed toward a target. The light is reflected by the target back to the sensor's position-sensitive device (PSD) as the receiver element

Bauform LG5/LG10, Laser-Abstandssensor
Housing type LG5/LG10, Laser position sensor

Bauform Housing	H × B × T [mm] H × W × D [mm]	Messbereich Measuring range	Brennweite focal distance	Wiederholgenauigkeit/ Repeat accuracy
LG5A	55,3 × 20,2 × 82,3	45…60 mm	70 mm	≥ ± 0,01 %[2]
LG5A	55,3 × 20,2 × 82,3	45…60 mm	53 mm	≥ ± 0,01 %[2]
LG10	55,3 × 20,2 × 82,3	75…125 mm	180 mm	≥ ± 0,01 %[2]

[2] vom Erfassungsabstand/of sensing distance

Series LT3
The LT3 uses **pulsed time-of-flight technology** to achieve unsurpassed performance. The laser pulses one million times per second. The microprocessor records the time required for each pulse to travel to the target and back to the sensor. Every millisecond, one thousand pulse times are averaged and the according value is transferred to the output.
The sensor's long range enables it to detect very small parts or inconspicuous features, even when it is mounted well back from the hazards of a process.

Bauform LT3, Laser-Relexionslichtaster, Hohe Reichweite
Housing type LT3, diffuse mode laser sensor, long range

Bauform Housing	H × B × T [mm] H × W × T [mm]	max. Reichweite max. range	Wiederholgenauigkeit/ Repeat accuracy
LT3…	68,5 × 35,3 × 87	0,3…5 m	≥ ± 1 mm
LT3…LV[1]	68,5 × 35,3 × 87	0,5…50 m	≥ ± 5 mm

[1] mit Reflector/with reflector

MINI-ARRAY™

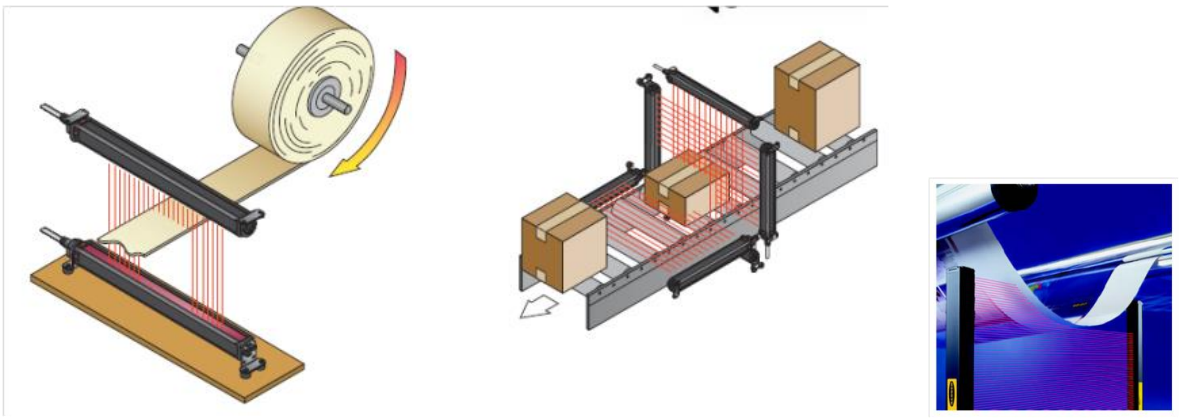


Figura 4. TURCK™. Industrial Automation.
Position and Displacement Sensors - Photoelectric sensors

2.2 Nuestro montaje óptico.

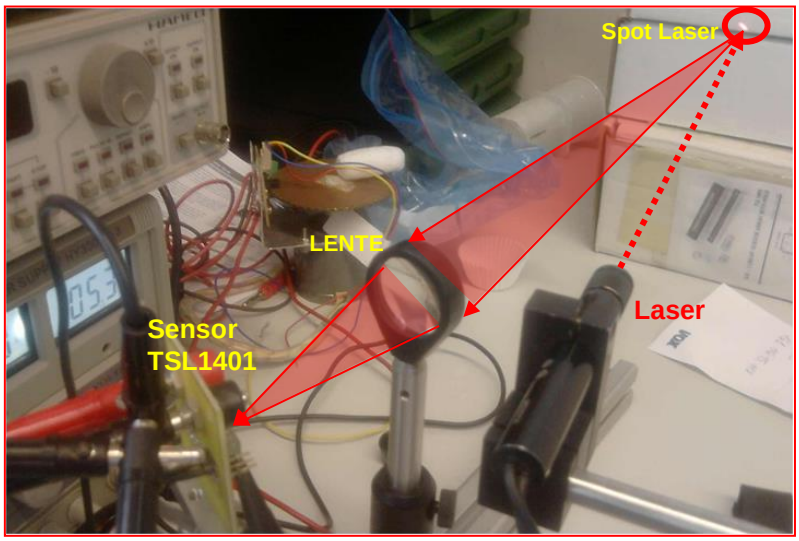


Figura 5. Fotografía del montaje óptico para la medida por triangulación óptica.
La lente recoge un cono de luz procedente del "spot". La cantidad de luz dependerá del diámetro.

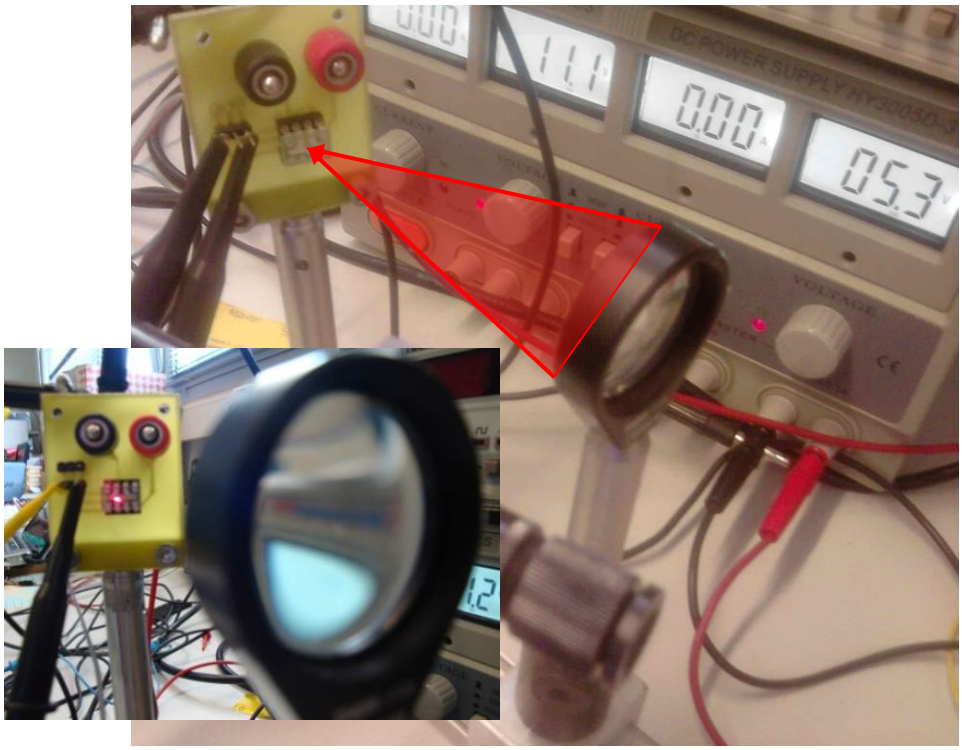


Figura 6. La lente focaliza el spot sobre el sensor.

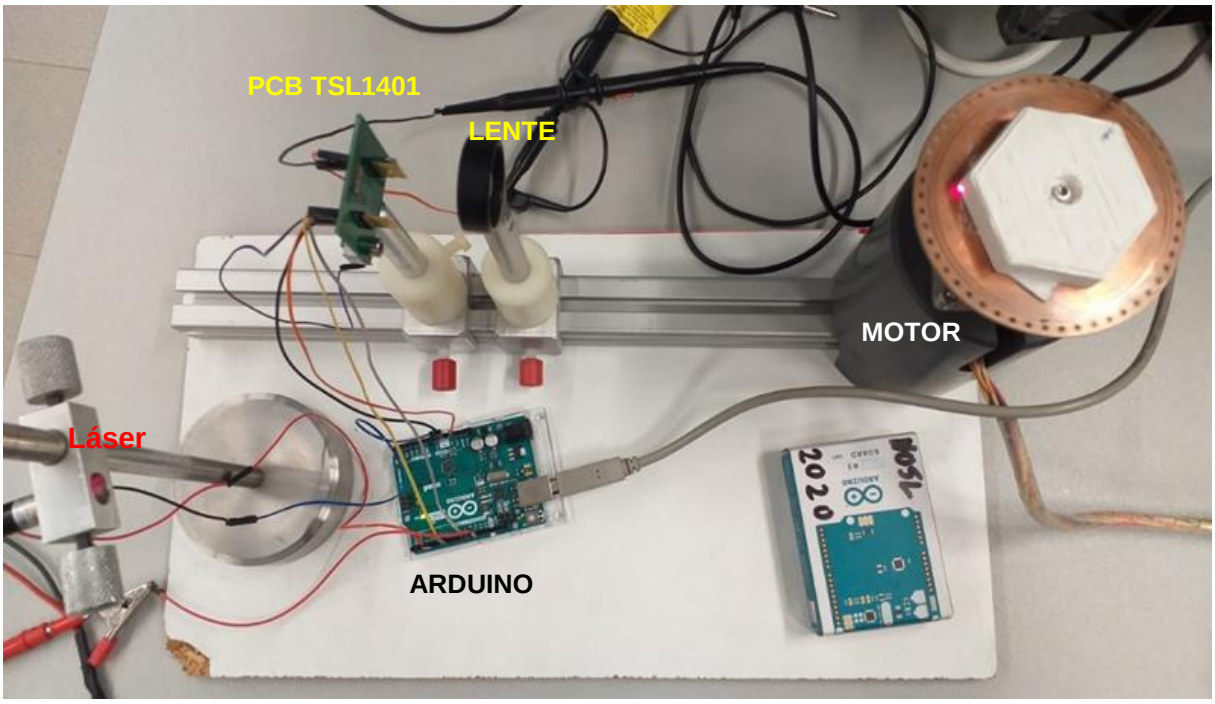


Figura 6b. Vista cenital del montaje.

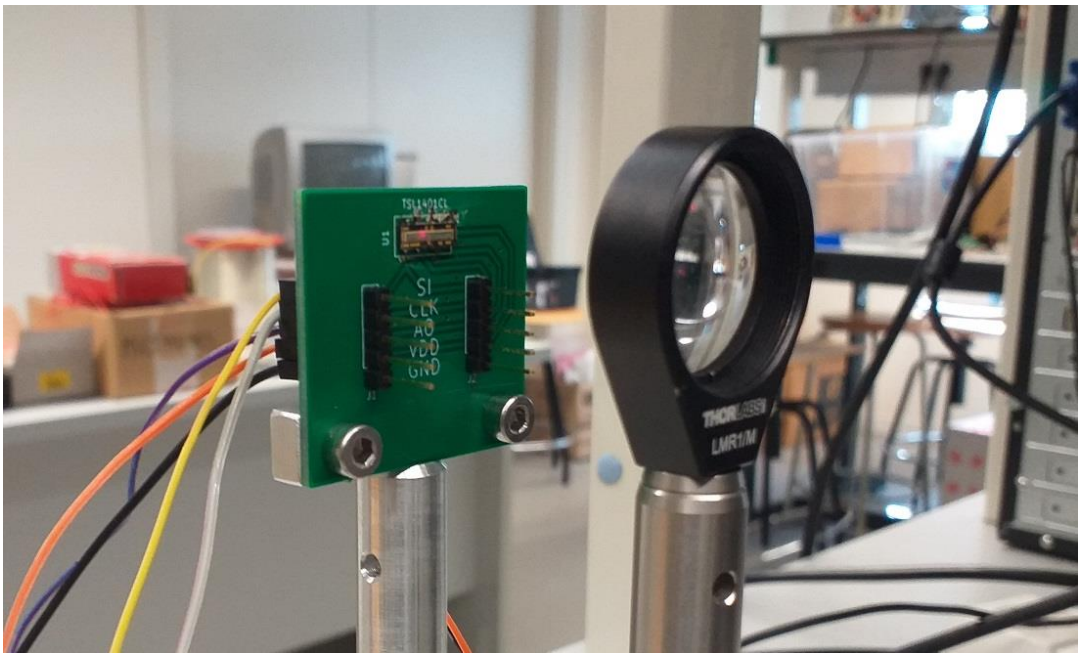


Figura 6c. Detalle lente más sensor con el spot-láser.

3. El array lineal de 128 fotodetectores TSL1401.

El TSL1401 es un array de 128 fotodetectores, cuyos pixels miden 63.5 μm (H) x 55.5 μm (W) con una separación entre centros de 63.5μm.

La figura 7 muestra el detalle de los pines del sensor lineal TSL1401, su función y el "timing" de las señales de control. La señal SI define el inicio de la lectura secuencial. El tiempo entre dos señales "**SI**" definen el "**tiempo de integración**". Cada pulso "**CLK**" define la salida secuencial de uno de los 128 pixels por el terminal "**AO**".

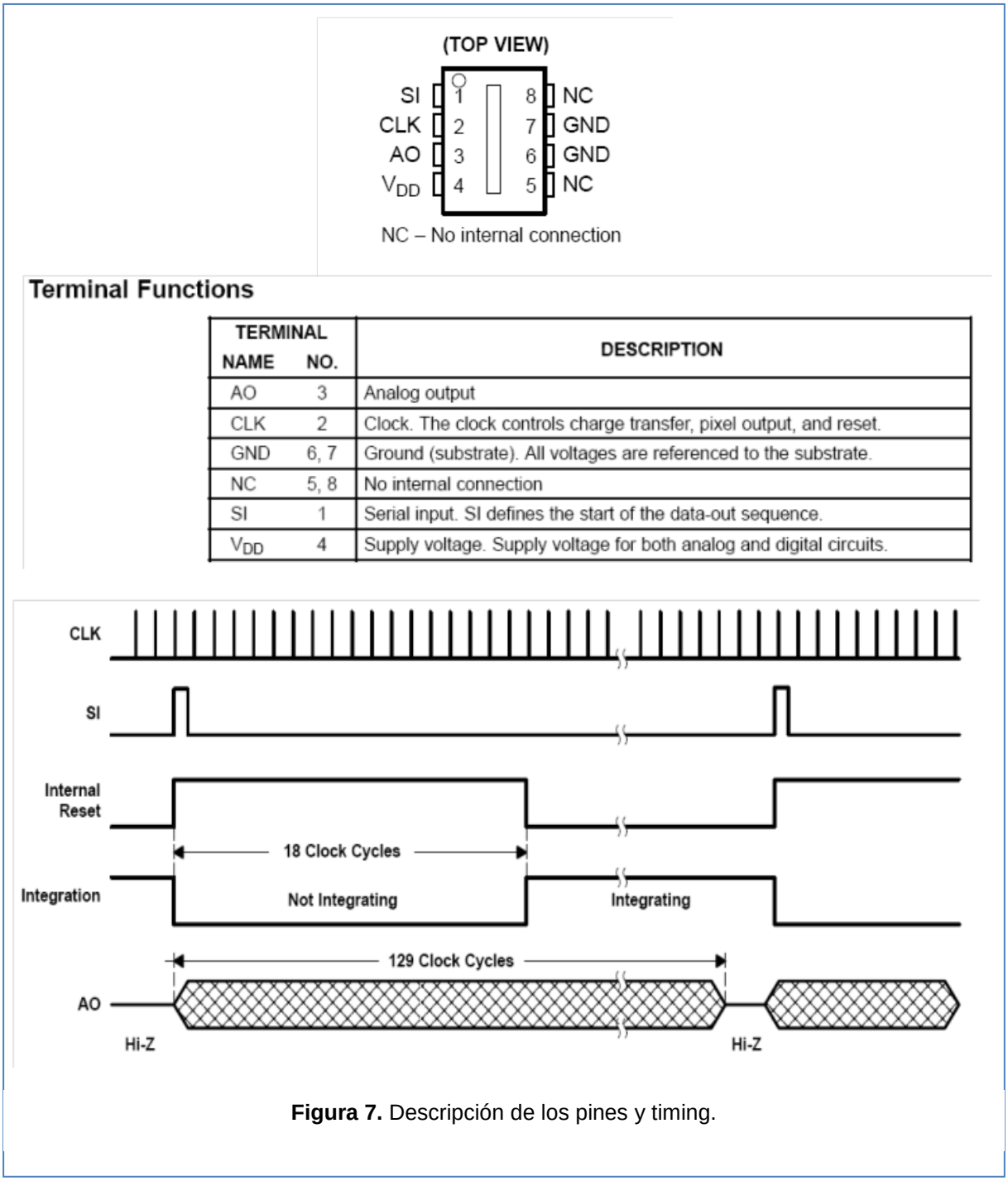


Figura 7. Descripción de los pines y timing.

En la figura 8 se muestran las condiciones de trabajo recomendadas.

Recommended Operating Conditions (see Figure 1 and Figure 2)

	MIN	NOM	MAX	UNIT
Supply voltage, V_{DD}	4.5	5	5.5	V
Input voltage, V_I	0		V_{DD}	V
High-level input voltage, V_{IH}	$V_{DD} \times 0.7$		V_{DD}	V
Low-level input voltage, V_{IL}	0		$V_{DD} \times 0.3$	V
Wavelength of light source, λ	400		1000	nm
Clock frequency, f_{clock}	5		2000	kHz
Sensor integration time, t_{int}	0.0645		100	ms

Figura 8. Valores recomendados.
El TSL1401 se alimenta a 5V

- La tensión de alimentación debe ser **V_{DD}=5V**.
- Respecto al valor digital de entrada, el valor mínimo permitido para **HIGH** es 0.7*5 =3.5. No obstante, las señales de control generadas por el PIC a 3.3V controlan perfectamente el dispositivo.

Finalmente, la figura 9 es una fotografía del PCB que utilizaremos con el TSL1401, con los dos pines de control (CLK y SI), el de salida (OUT), y los conectores de polarización (GND y V_{DD}).

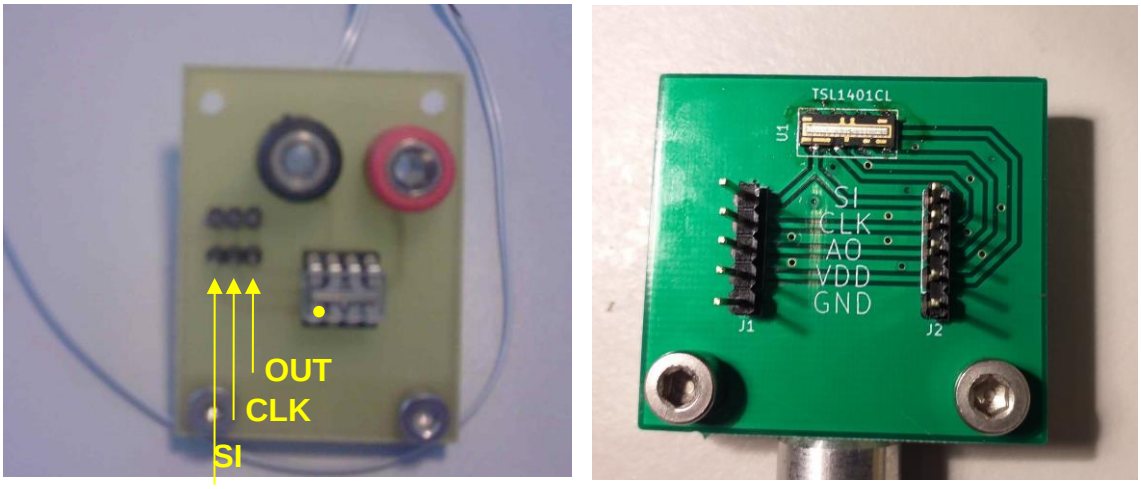


Figura 9. PCB conteniendo el TSL1401.

El microcontrolador.

La figura 10 muestra el diagrama de la placa ARDUINO - UNO.

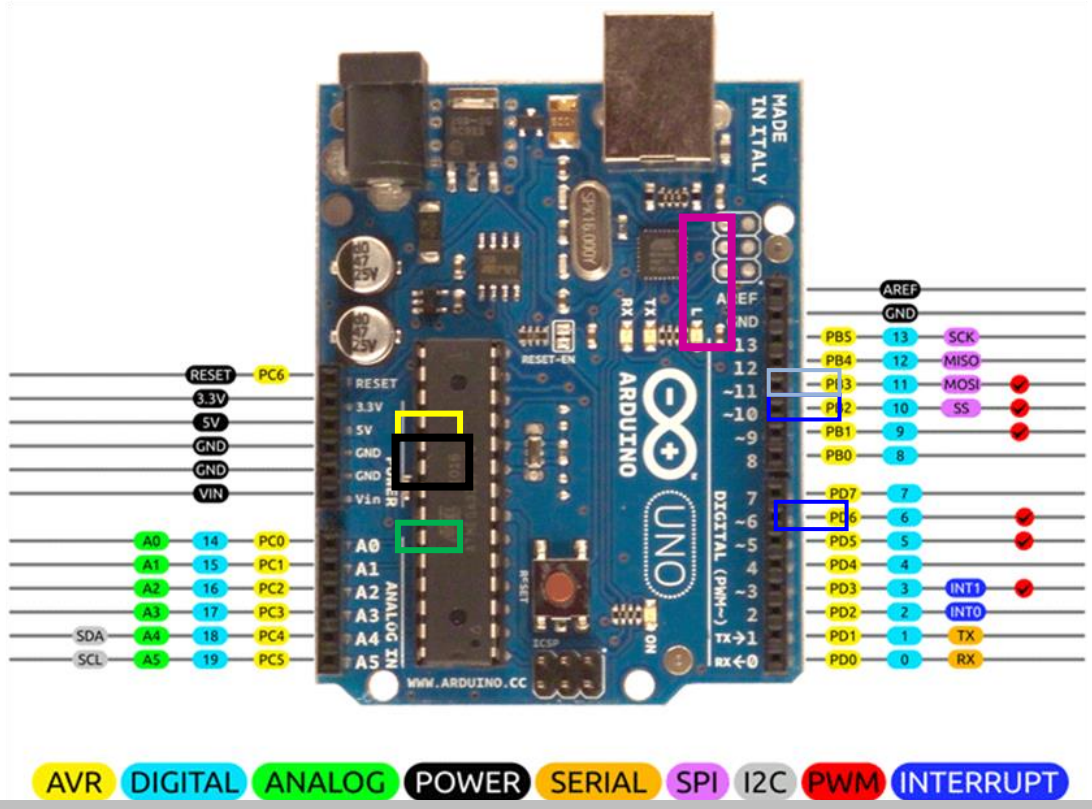
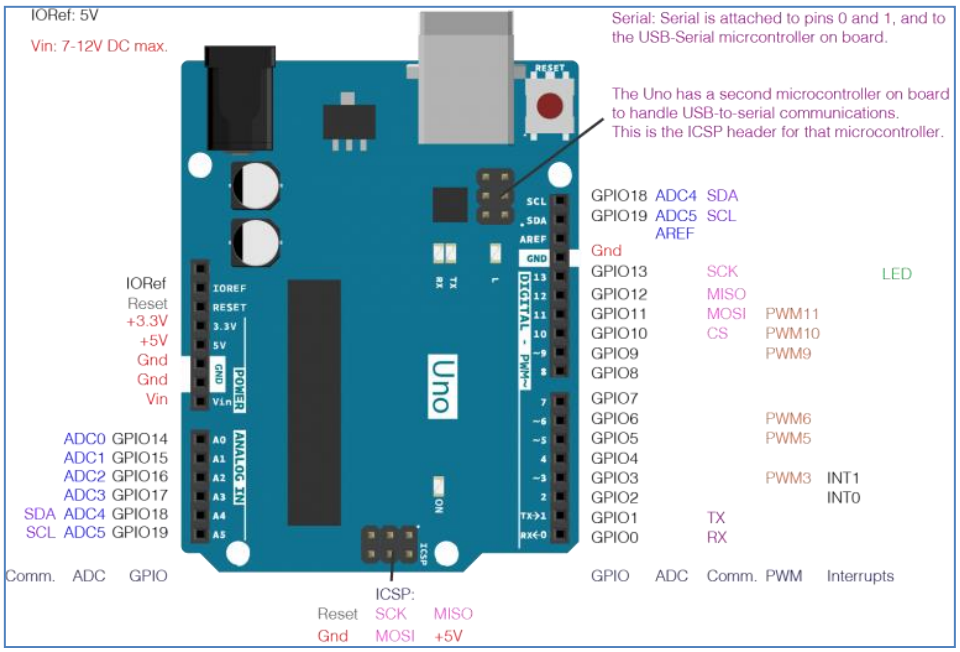


Figura 10. Detalle de los pines a utilizar (opcionales) de ARDUINO,
Amarillo = polarización; Negro = GND; azul = control (SI, CLK); Cian = motor
Verde = ADC; Magenta = comunicaciones, TX.



4. Nivel de programación 1: Generación de las señales de control del TSL1401.

- El TSL1401 se alimenta a V_{DD}=5V, tensión que podemos obtener de la placa ARDUINO

- Se debe seleccionar dos pines de salida para generar las señales SI y CLK.
- Inicialización de los pines:

```
#define TSL1401_ST 6
#define TSL1401_CLK 10
#define TSL1401_VIDEO A0
```

- La figura 11 es un detalle de la secuencia de inicio START

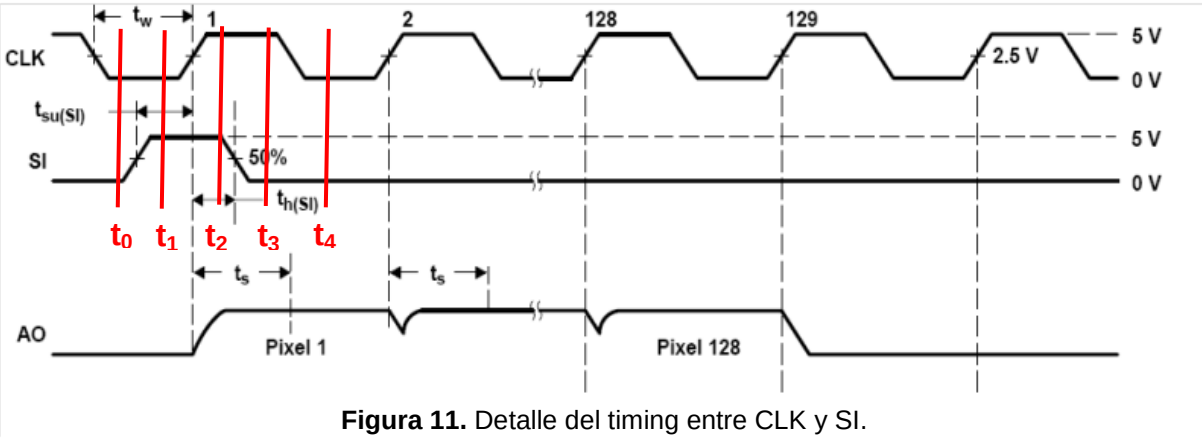


Figura 11. Detalle del timing entre CLK y SI.

Para generar la **secuencia START** definiremos la secuencia temporal de los valores de las señales SI y CLK en los instantes de tiempo t_0, t_1, t_2, t_3, t_4 y t_5 tal y como aparecen en la figura 11.

	CLK (pin 10)	SI (pin 6)
t_0	LOW	LOW
t_1	?	?
t_2	?	?
t_3	?	?
t_4	?	?
t_5	?	?
t_6	?	?

- **Programa principal**. Debe encargarse de generar las señales de control necesarias para gobernar el Array de fotodetectores. El esqueleto principal del programa puede consistir en un bucle de repetición infinita como el que a continuación se detalla:

```
#define TSL1401_ST 6
#define TSL1401_CLK 10
#define TSL1401_VIDEO A0
#define Pin_Motor 11

void setup()
 {
  pinMode(Pin_Motor, OUTPUT);
  pinMode(TSL1401_ST, OUTPUT);
  pinMode(TSL1401_CLK, OUTPUT);
  pinMode(A0,INPUT);
}

void loop()
  {
  // START SEQUENCE
  digitalWrite(TSL1401_ST,???); digitalWrite(TSL1401_CLK, ???);  delayMicroseconds(2);
  digitalWrite(TSL1401_ST, ???); digitalWrite(TSL1401_CLK, ???); delayMicroseconds(2);
  digitalWrite(TSL1401_ST, ???); digitalWrite(TSL1401_CLK, ???); delayMicroseconds(2);
  digitalWrite(TSL1401_ST, ???); digitalWrite(TSL1401_CLK, ???); delayMicroseconds(2);
  digitalWrite(TSL1401_ST, ???); digitalWrite(TSL1401_CLK, ???);  delayMicroseconds(2);
  //
  for(int i=0;i< nPixel;i++)
     {
     digitalWrite(TSL1401_CLK, ???);  delayMicroseconds(5);
     digitalWrite(TSL1401_CLK, ???);  delayMicroseconds(5);
     }

  // SE REPITE LA SECUENCIA DE START
  // CUANDO SE TRANSMITE EN SERIE EL TIEMPO DE INTEGRACION SIGUE TRABAJANDO
  digitalWrite(TSL1401_ST,???); digitalWrite(TSL1401_CLK, ???);  delayMicroseconds(2);
  digitalWrite(TSL1401_ST, ???); digitalWrite(TSL1401_CLK, ???); delayMicroseconds(2);
  digitalWrite(TSL1401_ST, ???); digitalWrite(TSL1401_CLK, ???); delayMicroseconds(2);
  digitalWrite(TSL1401_ST, ???); digitalWrite(TSL1401_CLK, ???); delayMicroseconds(2);
  digitalWrite(TSL1401_ST, ???); digitalWrite(TSL1401_CLK, ???);  delayMicroseconds(2);
}
```

Figura 12. Líneas generales del código para controlar el TSL1401

Con este código se puede comprobar el funcionamiento del array de fotodetectores. La figura 13-izquierda muestra las capturas de las señales de control, SI y CLK. La figura 13-derecha muestra la salida analógica del array de 128 pixeles, cuando incide un láser sobre el array de detectores. Se toma como referencia la señal SI (amarillo) para el sincronismo.

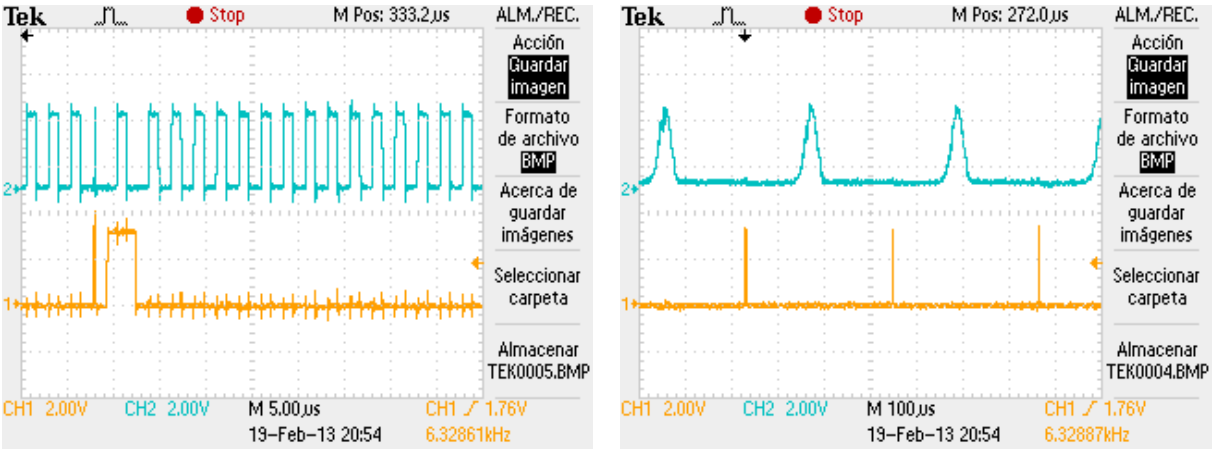


Figura 13. Izquierda: azul = CLK, amarillo = START
Derecha: azul = sensor output, amarillo = START

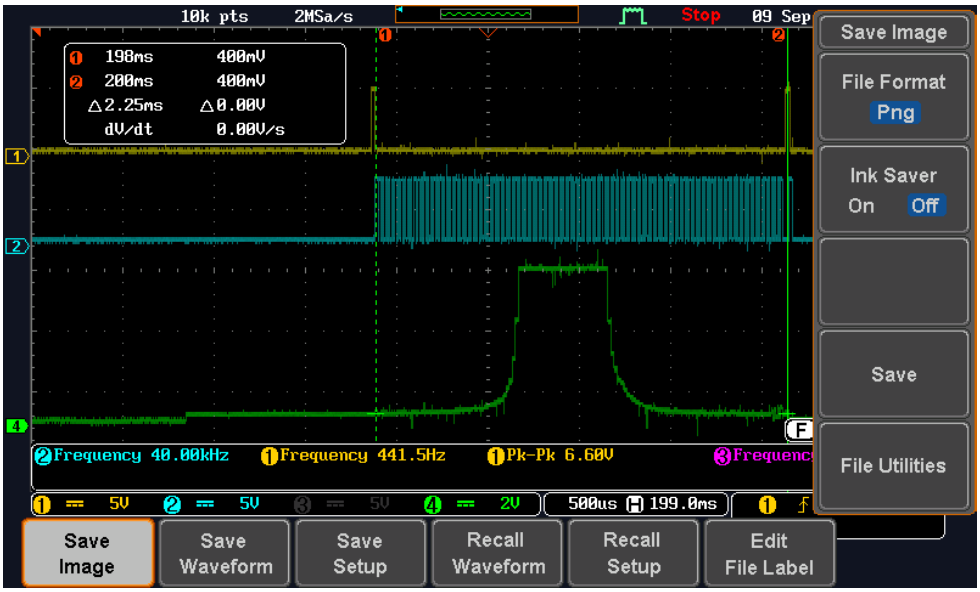


Figura 14. Amarillo = START; Azul = CLK ; Verde = sensor output.
Δt_{TOTAL} array =2.25ms

La figura 14 muestra la salida del sensor TSL1401, que dura alrededor de **Δt_{TOTAL} =2.25ms**. En este tiempo se incluyen los ***'delayMicroseconds'*** utilizados en cada actualización de los pines, aunque no es un tiempo excesivo. Además el control de un paso de motor, que se comentará en la siguiente sección.

La figura 15 es un detalle de la señal CLK que controla la salida del sensor. Es bastante simétrica, **Δt_{PIXEL} ≈21µs**. El tiempo total es **$\Delta t_{TOTAL}=128\cdot\Delta t_{PIXEL}$ ≈ 2.688ms**, que es similar el medido directamente en el ociloscopio en la figura 14.

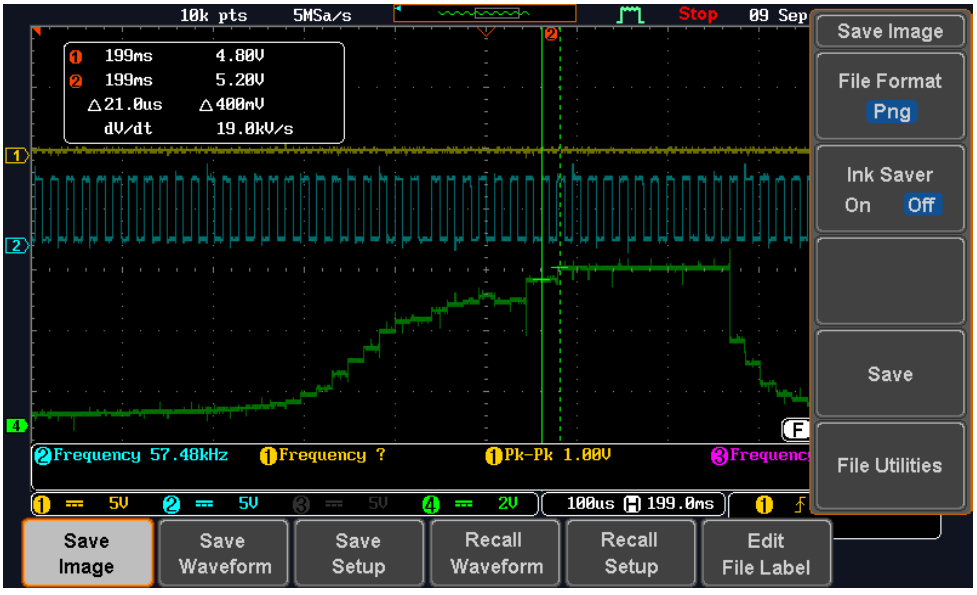


Figura 15. Amarillo = START; Azul = CLK ; Verde = sensor output
Δt_{PIXEL} ≈21µs

5. Nivel de programación 2: motor paso-a-paso.

El objetivo global es rotar un objeto con ayuda de un motor, y medir la distancia del objeto, concretamente la del spot del láser sobre el objeto, al sistema óptico de medida.

Añadiremos una prestación adicional al código anterior del microcontrolador. Se utilizará uno de los pines digitales de salida para generar una señal cuadrada y controlar el movimiento de nuestro motor paso-a-paso. En la figura 12 se ha definido el **Pin_Motor=11**

El movimiento del motor se realizará antes de efectuar la lectura del sensor lineal de fotodetectores. Según como estén dispuestas las fases en cada una de las placas de control, **se puede necesitar una o dos secuencias HIGH-LOW** para generar un paso de motor que equivale a ángulo=1.8º Se debe añadir un tiempo de espera razonable para observar la rotación del motor.

El texto que se debe incluir antes de programar la lectura del ADC podría ser:

```
// MOVIMIENTO DEL MOTOR (un paso)
  digitalWrite(Pin_Motor,HIGH);
    delay(??) ;
  digitalWrite(Pin_Motor,LOW) ;
    delay(??) ;
  digitalWrite(Pin_Motor,HIGH);
    delay(??) ;
  digitalWrite(Pin_Motor,LOW) ;
    delay(??) ;
```

Figura 16. Código para mover un paso el motor.

La figura 17 muestra la tarjeta controladora del motor paso-a-paso.

1. Se tiene una entrada de **5V** para polarizar la electrónica digital de la placa.
2. La alimentación del motor será del orden de **V_{motor}=10V**
3. Ambas tensiones las obtendremos de una fuente de alimentación externa.
4. El conector BNC es la señal pulsada 0-5V generada con ARDUINO. Utilizar en este caso un cable BNC-banana/cocodrilo para conectarse a ARDUINO.

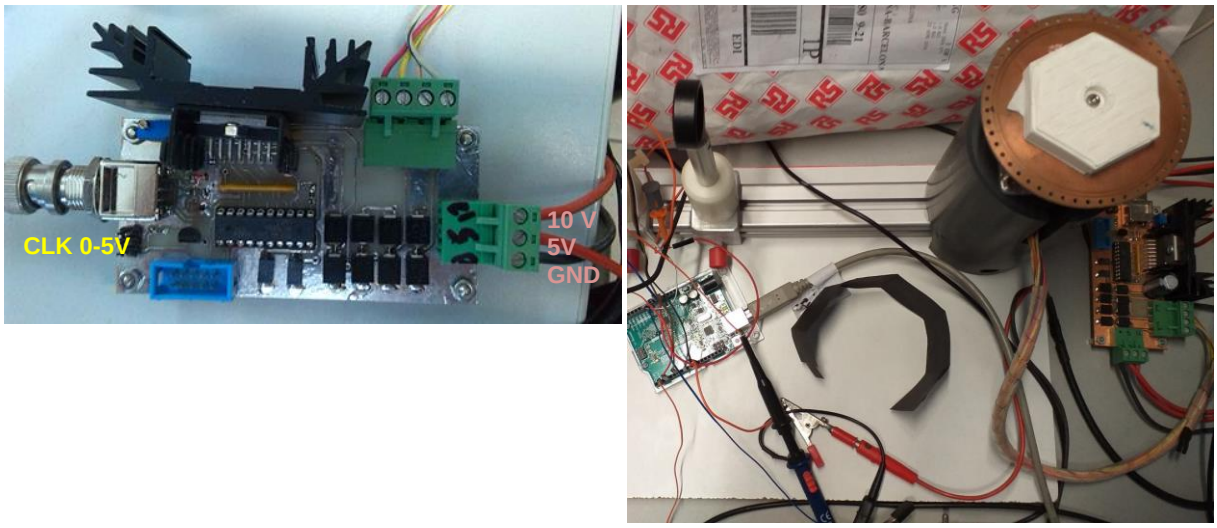


Figura 17. Tarjeta controladora y motor paso-a-paso

7. Nivel de programación 3: adquisición de la lectura del sensor con el ADC

7.1 Objetivo:

En cada paso de motor, llamar al **ADC** para adquirir los 128 valores que entrega el array de fotodetectores. Estos valores se almacenarán en un vector

7.2 Descripción:

```
int vector[128]; // Vector para guardar la variable analógica
….
…..

 //
 for(int i=0;i< nPixel;i++)
    {
    digitalWrite(TSL1401_CLK,HIGH);  delayMicroseconds(ESPERA);
    digitalWrite(TSL1401_CLK,LOW);  delayMicroseconds(ESPERA);
    vector[i] = analogRead(A0) ;
    }
```

Figura 18. Líneas de código para llamar al ADC.

NOTA IMPORTANTE

El tipo de programación descrito en el cuadro de la figura 12 podría haberse realizado generando una señal PWM para producir la señal CLK y una señal SI para inicializar la integración. En este caso:

- La señal de salida AO del sensor estaría sincronizada con el PWM.
- La llamada al ADC debería realizarse también de forma síncrona con el PWM, asumiendo que el tiempo de conversión debe estar contenido en un ciclo de la señal CLK-PWM.
- Nos obligaría a configurar los registros del ADC para llevar a cabo la digitalización de la señal.

Creemos que la programación programando los GPIO es algo más sencilla que la que se acaba de proponer.

ENTREGA REPORT BÁSICO:

- **Captura del osciloscopio mostrando las señales SI y CLK**
- **Captura del osciloscopio mostrando las señales SI y OUT**
  - **Con el spot del láser**
  - **Con una cartulina negra que tape medio sensor: "detector de bordes"**

- **Captura del osciloscopio de SI y OUT con LUZ AMBIENTE.**
- **Captura del osciloscopio de SI y OUT con EL SENSOR HACIA ABAJO.**
  - **Indicar en este caso el valor Vout y la frecuencia CLK.**
  - **Aumentar el tiempo de integración para obtener un valor Vout mayor. Valor de Vout y frecuencia CLK.**

- **Captura del osciloscopio con la transmisión de los primeros bytes a través de la UART, mostrando la cabecera.**

- **Captura del osciloscopio con la salida del sensor (CANAL1) y con la señal UART (CANAL2).**

Nivel de programación 4: configuración/transmisión con UART

8.1 Inicialización de la UART.

```
Example Code:
void setup() {
  //Initialize serial and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB
  }
}
void loop() {
  //proceed normally
}
```

Figura 19. Líneas de código de inicialización de la UART.
https://www.arduino.cc/reference/en/language/functions/communication/serial/

8.2. Transmisión con la UART.

Tras terminar de almacenar los 128 valores del array TSL1401 se transmiten los 128X2 bytes.

```
// TRANSMISION DE LA CABECERA
  Serial.write(255) ;
  Serial.write(0) ;

  // TRANSMISION DE LOS 128 PIXELS
  for(int i=0;i< nPixel;i++)
    {
    Serial.write(highByte(vector[i])) ;
    Serial.write(lowByte(vector[i]))  ;
    }
  delay(5) ;
```

Figura 20. Líneas de código para la **transmisión de los 128x2 bytes** previamente almacenados.

Serial.write()

Description

Writes binary data to the serial port. This data is sent as a byte or series of bytes; to send the characters representing the digits of a number use the print() function instead.

Syntax

Serial.write(val)
Serial.write(str)
Serial.write(buf, len)

Parameters

Serial: serial port object. See the list of available serial ports for each board on the Serial main page.
val: a value to send as a single byte.
str: a string to send as a series of bytes.
buf: an array to send as a series of bytes.
len: the number of bytes to be sent from the array.

Returns

write() will return the number of bytes written, though reading that number is optional. Data type: size_t.

Example Code

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

void loop() {
  Serial.write(45); // send a byte with the value 45

  int bytesSent = Serial.write("hello");  //send the string "hello" and return the length of the string.
}
```

Figura 21. Descripción de la función Serial.write() de ARDUINO.
https://www.arduino.cc/reference/en/language/functions/communication/serial/

8.3 Uso de la UART (OPCIONAL)

Se comprobará la correcta programación de la UART. Para ello se programa una transmisión secuencial de números enteros en orden creciente. El resultado debe ser equivalente a lo que se muestra en la figura 22. Se observa que **la duración de un símbolo es de 69µs**, que corresponde a la velocidad de transmisión de **115200bps**. Observar el bit de START y el bit de STOP

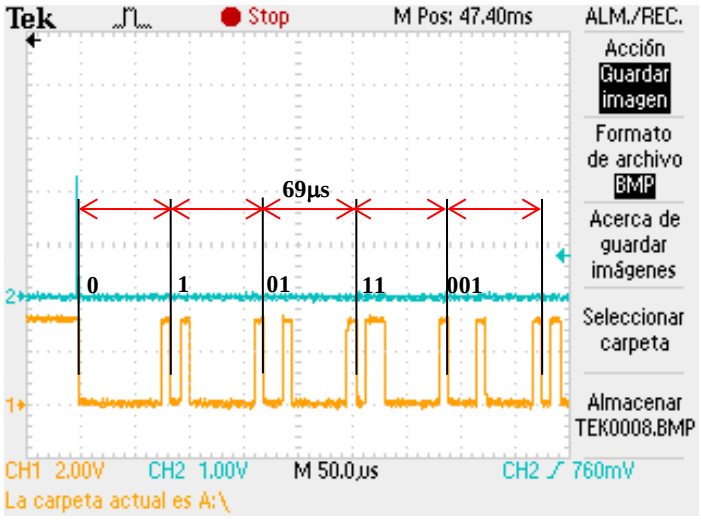


Figura 22. Azul: pulso START del sensor TSL1401
Amarillo: transmisión serie de números enteros en orden creciente: 0,1,2,3,4,….

En la figura 23, en verde se tiene la salida del sensor, que coincide con la entrada del ADC, y en magenta, después de la adquisición del perfil, la transmisión serie de los primeros bytes. Se observa la cabecera “FF”+”0”.
El tiempo necesario para la transmisión serie de la información es de [2+128*2]*70µs =**18060µs ≈ 18ms**, que, junto a unas demoras adicionales introducidas por software, da un total de unos 22ms.

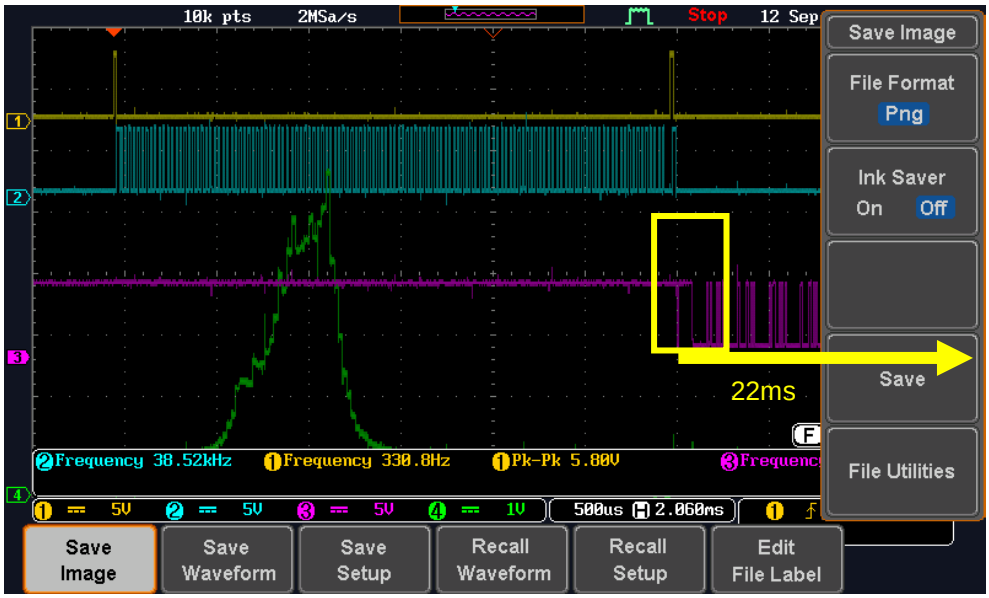


Figura 23. Amarillo = START; Azul = CLK ; Verde = sensor output; Magenta = UART
Captura del sensor y transmisión de los primeros pixels (2bytes/pixel).

En la figura 24, se observa toda la ráfaga de la UART y los 22ms de duración.

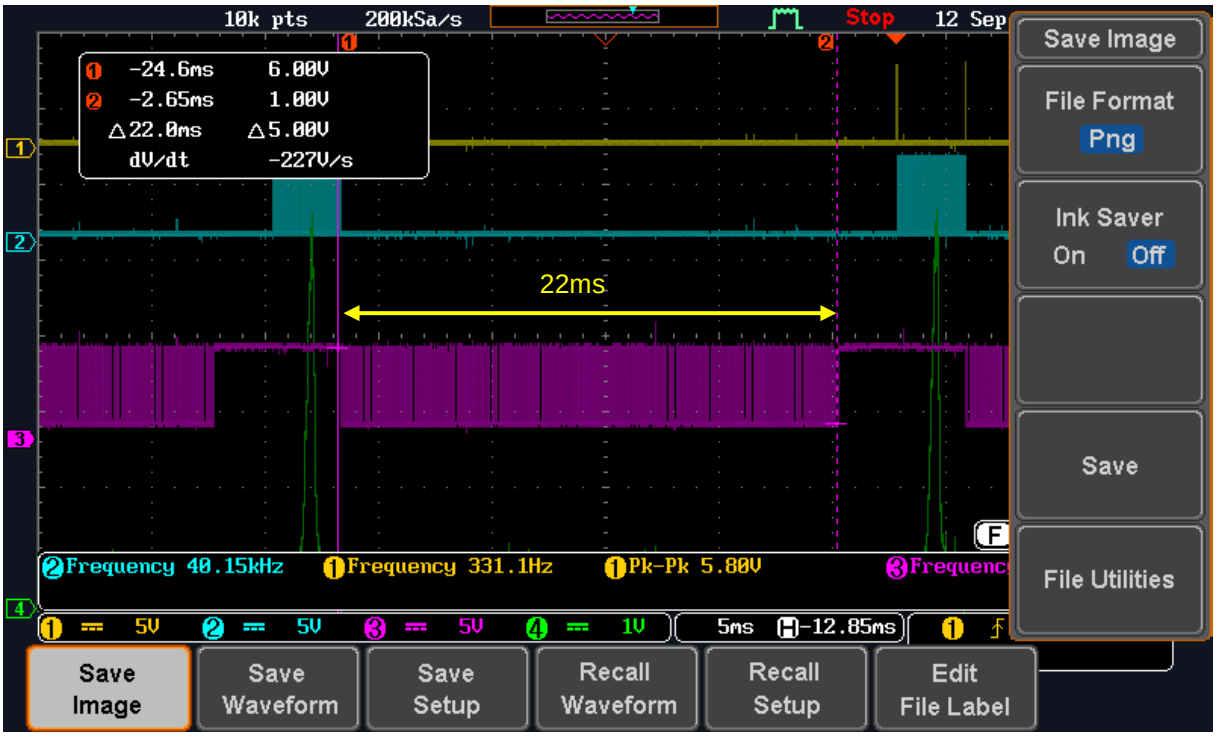


Figura 24. Detalle de los tiempos totales.
Amarillo = START; Azul = CLK ; Verde = sensor output (<3ms); Magenta = UART (22ms)

8. Nivel de programación 5: recepción de datos con MATLAB.

9.1 Inicialización de las comunicaciones.

La primera tarea para realizar es localizar el número de puerto correspondiente a ARDUINO. Para ello acceder a "**administrador de Dispositivos**" + "**Puertos (COM y LPT)**" de su ordenador y **buscar el número de puerto asignado** por WINDOWS®. También sabemos el número de puerto desde el momento en que se programa ARDUINO.

En la figura 25 se describen las instrucciones básicas en MATLAB para configurar el puerto serie.

```
Puerto = 'COM26' ;
%% borrar posible conexión previa
delete(instrfind({'Port'},{Puerto}));
%arduino = serial(Puerto,'BaudRate',115000,'Terminator','CR/LF');
arduino = serial('COM26') ;
set(arduino,'BaudRate',115200) % speed of data
set(arduino,'InputBufferSize',(2+128*2)*1 ) ; % Size of the buffer
set(arduino,'InputBufferSize',(2+128*2)*1 ) ; % Size of the buffer
set(arduino,'DataBits',8) % define number ob bits
set(arduino,'Parity','none'); % No parity
set(arduino,'StopBits',1); % number of bit stops
```

Figura 25. Código MATLAB para configurar la conexión serie

9.2 Lectura de datos.

Para evitar conflictos o pérdida de tramas enviadas por ARDUINO, sería interesante que MATLAB solicitase la ráfaga de 128 pixels. Para ello se transmite un número o carácter que deberá reconocer ARDUINO. En este caso **transmitimos el número entero 2**

```
fwrite(arduino,2) ;        % Escritura de MATLAB en el puerto serie
```

En ARDUINO, dentro de loop() se deberá estar chequeando continuamente el puerto serie, y actuar cuando se reciba este entero = 2.

```
if(Serial.available()>0)    // Comando en ARDUINO para leer el puerto serie
```

Con este ejemplo si ARDUINO recibiese un “2” debería ejecutar lo descrito en las secciones anteriores: secuencia START, 128 CLKs y 128 llamadas al ADC

Una propuesta de código en MATLAB sería:

1. Enviar desde el PC un carácter solicitando lectura de “frame”

2. Leer dos bytes consecutivos. En la sección 8.3 hemos utilizado “FF”+”0”
   - %Leemos un primer byte
   - aux1 = fread(arduino,1) ;
   - % Leemos un segundo byte
   - aux2 = fread(arduino,1) ;
3. Si estos coinciden secuencialmente con ‘FF’ i ‘00’ se trata de una cabecera y se procede a leer los siguientes 128x2 bytes.

EJEMPLO:

```
  fwrite(arduino,2) ;
  flushinput(arduino) ;
  flushoutput(arduino) ;
  %Leemos un primer byte
  aux1 = fread(arduino,1) ;
  % Leemos un segundo byte
  aux2 = fread(arduino,1)  ;
%
  if aux1 == 255 && aux2 == 0
        frame = frame + 1 ;
        disp(sprintf('Head detected %d',frame)) ;
        for pixel=1:128
           byte_alto = fread(arduino,1) ;    % Primer byte
           byte_bajo = fread(arduino,1) ;    % Segundo byte
           Amplitud(pixel) = byte_bajo  + byte_alto*256 ;
       end
  end
  %
  clf ;
  plot(Amplitud,'*') ;
  xlabel('#pixel') ;    ylabel('Intensity') ;
  set(gca, 'FontSize',12, 'FontWeight','bold','YTick',[0 64 128 256 512 768 1024]);
  axis([0 128 0 1024]) ;
  grid minor
```

Figura 26. Líneas de código MATLAB para leer la UART con los datos de ARDUINO

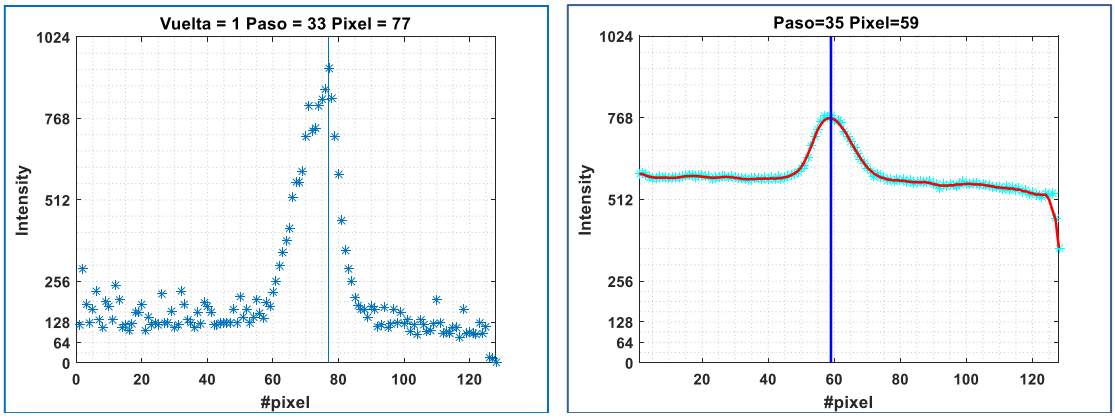


Figura 27. Captura por MATLAB de una trama de 128 pixels y posición del pixel más iluminado. Derecha: Cian = señal recibida. Rojo = señal suavizada

La figura 28-izquierda representa la captura de una figura con forma hexagonal. Cada lado del hexágono genera cada uno de los lóbulos de la figura, tanto en la representación cartesiana como la polar, figura 28-derecha.

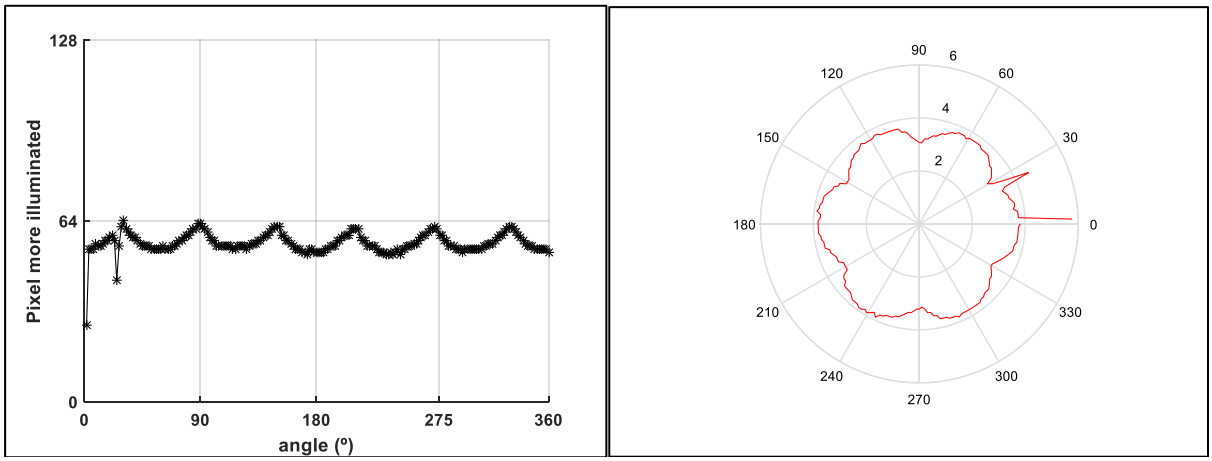


Figura 28. Representación cartesiana (izquierda) y polar (derecha) del pixel más iluminado para cada uno de los pasos de motor. **200pasos x1.8º = 360º**. La figura original es un hexágono.

9.3 Corrección / calibrado.

El sistema óptico no es lineal y debe hacerse un **calibrado previo del pixel más iluminado vs distancia**. Para ello, rotando manualmente el objeto, se mide el pixel más iluminado para las distancias máxima y mínima del objeto a escanear.

$$D_{min} \Rightarrow pixel_{min}$$
$$D_{max} \Rightarrow pixel_{max}$$

No obstante, como nuestro rango de distancias es pequeño supondremos linealidad. Con estos cuatro valores obtener la ecuación de una recta, D = f(pixel), que proporcione la distancia en función del pixel más iluminado.

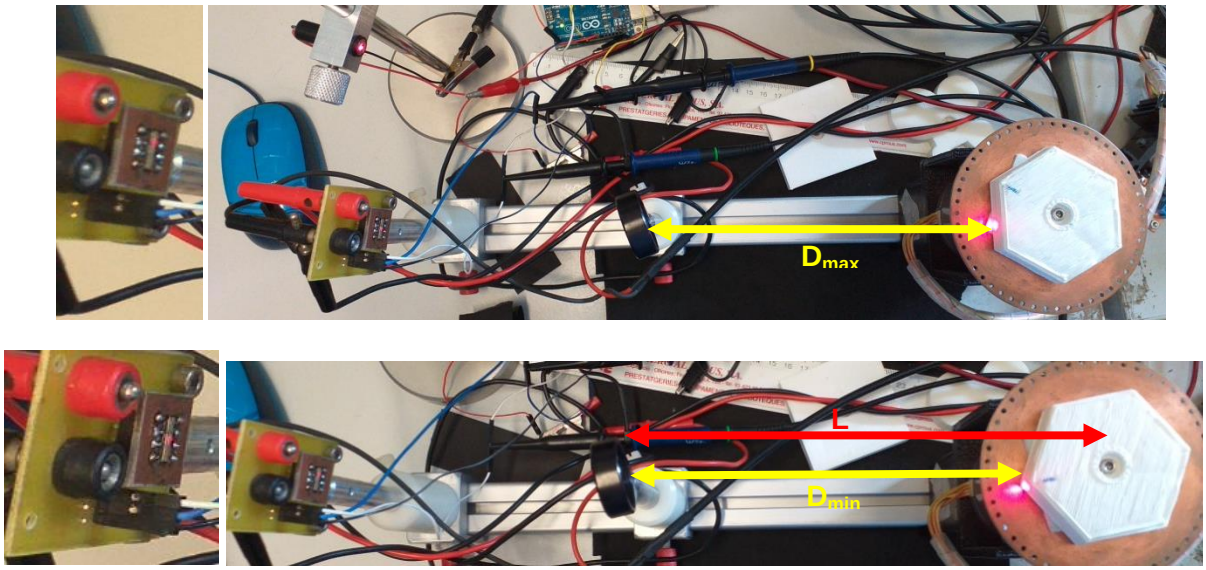


Figura 29. Medida de [D_{min}, $pixel_{min}$] y [D_{max},$pixel_{max}$]

Tras corregir los valores del pixel más iluminado, la figura 30 muestra el resultado final aproximado para la figura hexagonal escaneada:

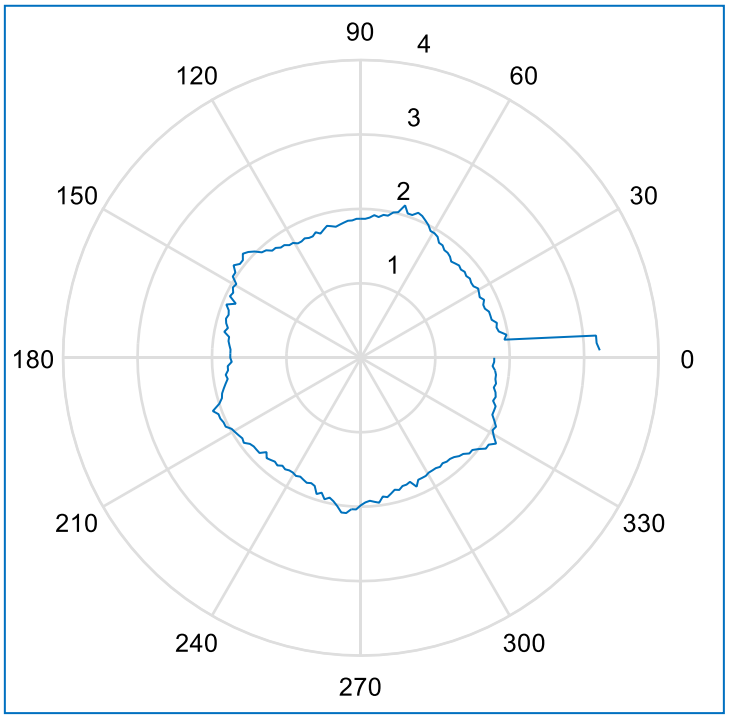
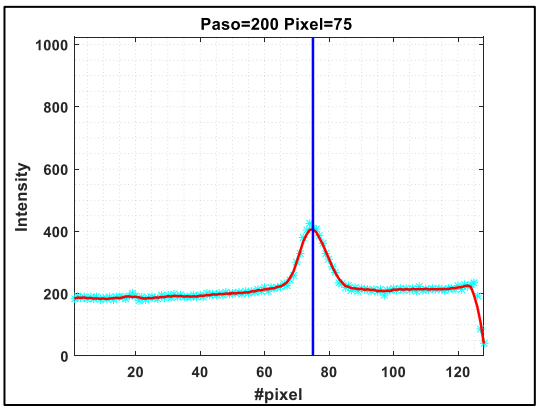
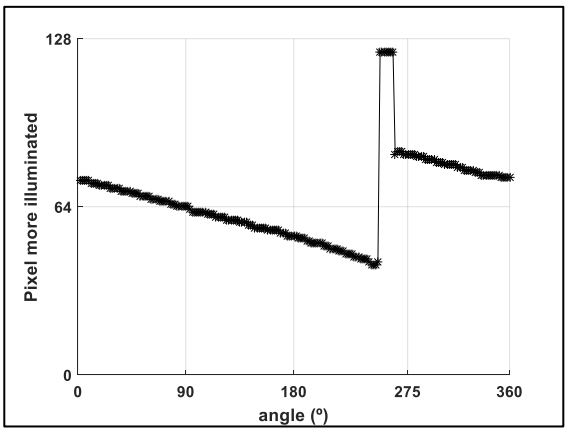


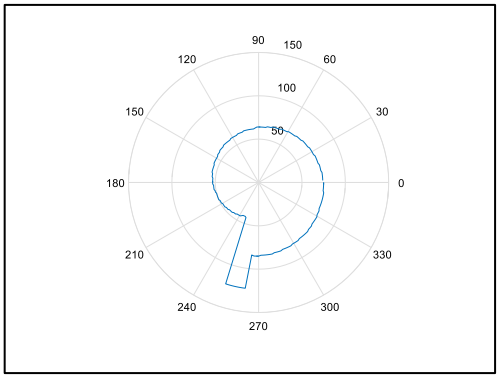
Figura 30. Corrección a distancias desde el eje del motor del objeto de la figura 31.

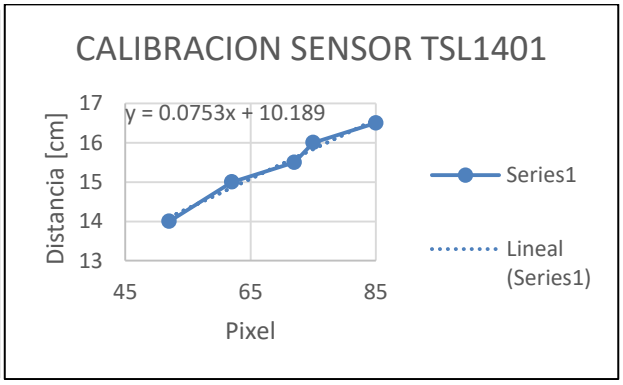

STOP

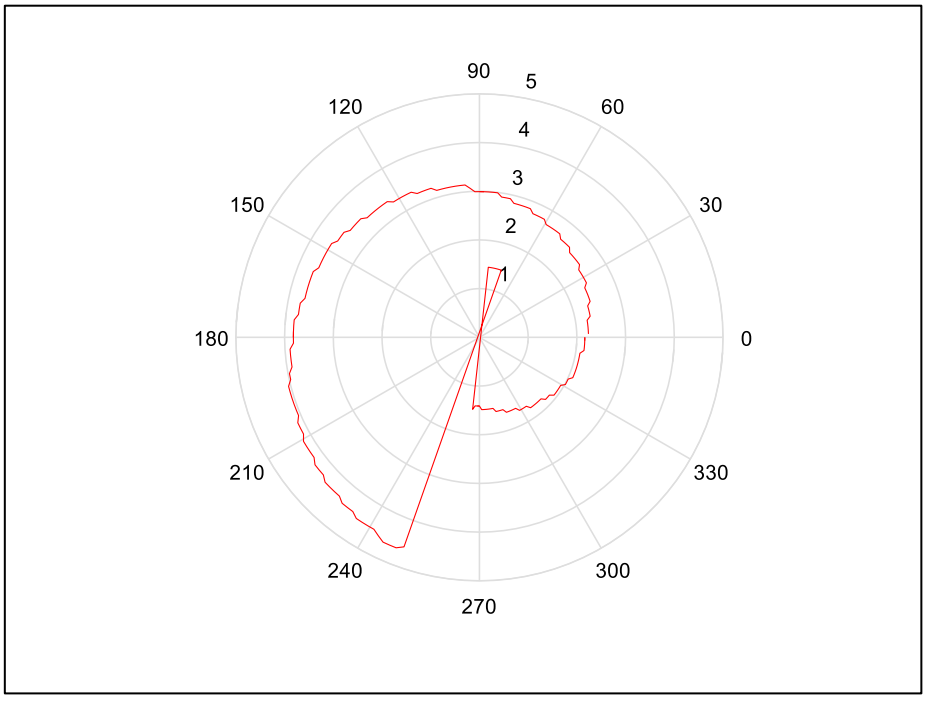
TRIANGULACIÓN 6 NOVIEMBRE 2020











8. APÉNDICE: adquisición de la lectura del sensor con el ADC.

8.1 Objetivo:

1. Configurar el ADC del ARDUINO-UNO.
2. En cada paso de motor, llamar al ADC para adquirir los 128 valores del array de fotodetectores.

7.2 General Descripción:

- The ADC is connected to an 8-channel Analog Multiplexer which allows eight single-ended voltage inputs constructed from the pins of Port A.
- The single-ended voltage inputs refer to 0V (GND).
- The analog input channel is selected by writing to the MUX bits in ADMUX.
- The ADC converts an analog input voltage to a 10-bit digital value through successive approximation. The minimum value represents GND and the maximum value represents the voltage on the AREF pin minus 1 LSB

Figure 24-1. Analog to Digital Converter Block Schematic Operation,

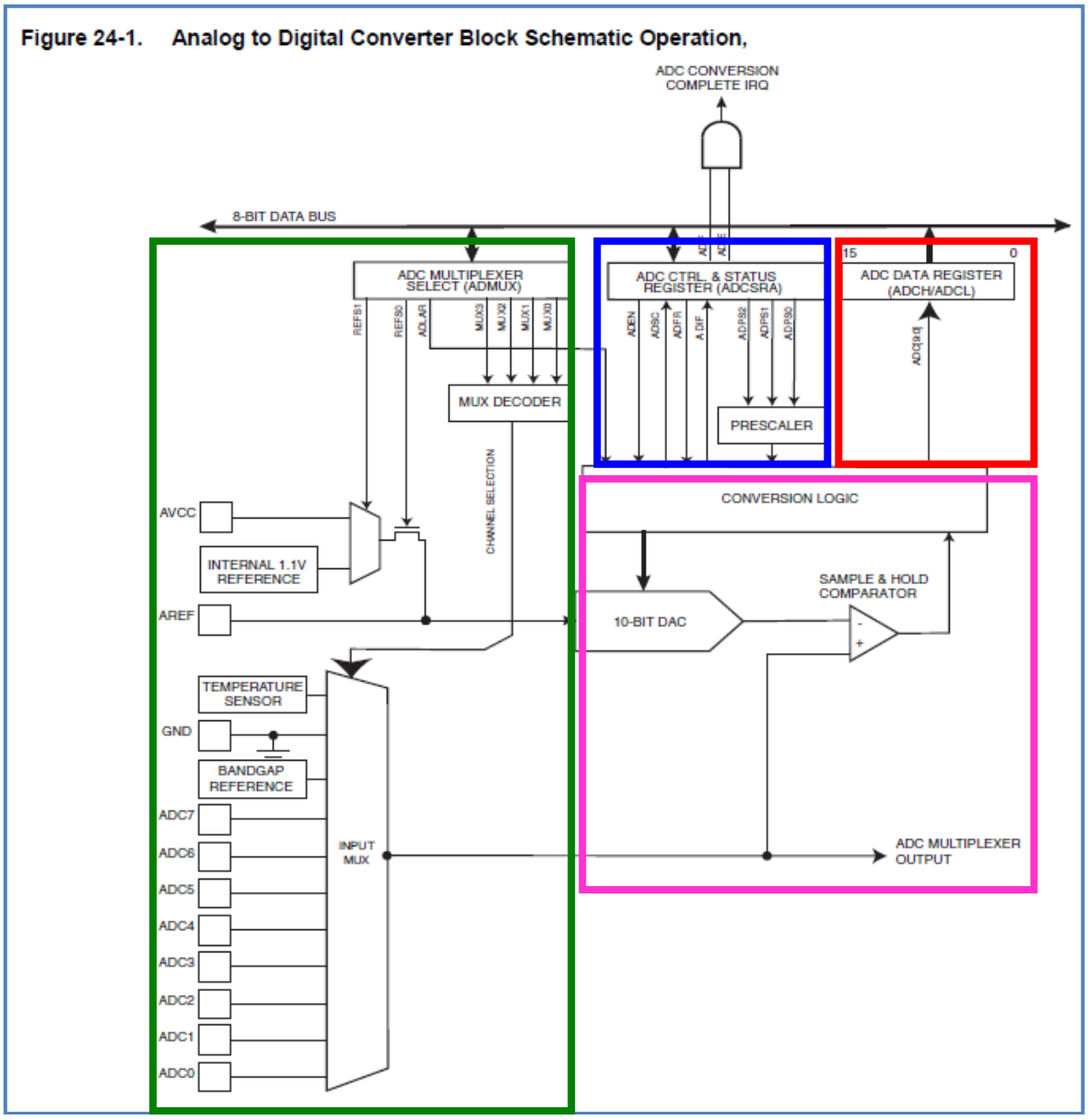


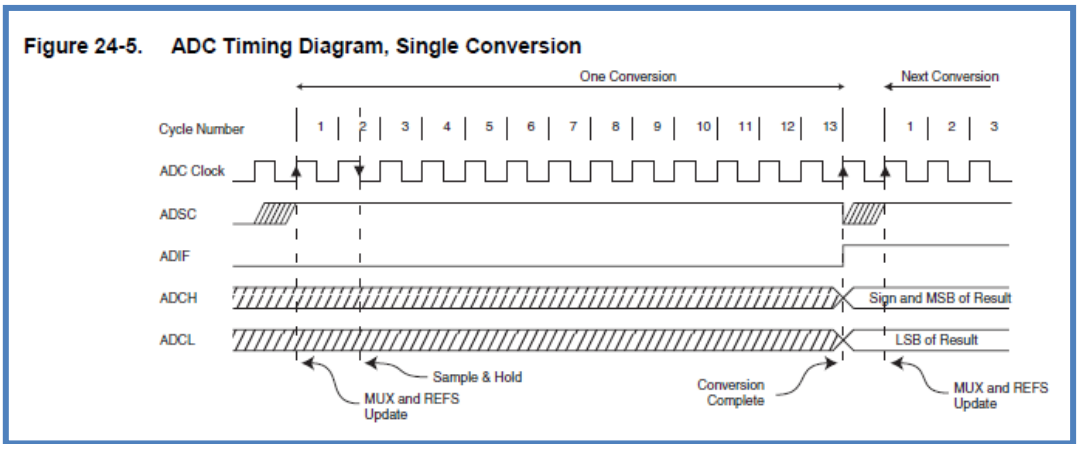
Figura 17. Esquema del ADC.

7.3 ADC Conversion.

- The ADC is enabled by setting the ADC Enable bit, ADEN in ADCSRA.
- The ADC generates a 10-bit result which is presented in the ADC Data Registers, ADCH and ADCL. By default, the result is presented right adjusted, but can optionally be presented left adjusted by setting the ADLAR bit in ADMUX.
- If the result is left adjusted and no more than 8-bit precision is required, it is sufficient to read ADCH.
- Otherwise,ADCL must be read first, then ADCH, to ensure that the content of the Data Registers belongs to the same conversion. Once ADCL is read, ADC access to Data Registers is blocked.

7.4 Preescaler:

- By default, the successive approximation circuitry requires an input clock frequency between 50kHz and 200kHz to get maximum resolution.
- The ADC module contains a prescaler, which generates an acceptable ADC clock frequency from any CPU frequency above 100kHz. The prescaling is set by the ADPS bits in ADCSRA.
- A normal conversion takes 13 ADC clock cycles. The first conversion after the ADC is switched on (ADEN in ADCSRA is set) takes 25 ADC clock cycles in order to initialize the analog circuitry.
- When a conversion is complete, the result is written to the ADC Data Registers (ADCL & ADCH), and ADIF is set.



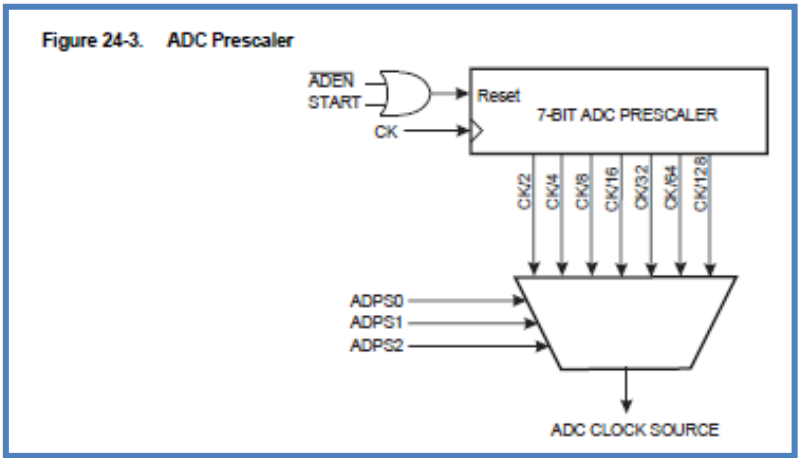


Figura 18. Arriba: Una conversión normal necesita 13 clocks del ADC.
Abajo: El preescaler controla el ADC-CLOCK.

7.5 Registros.

24.9.1 ADMUX – ADC Multiplexer Selection Register

Bit	7	6	5	4	3	2	1	0	
(0x7C)	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – REFS[1:0]: Reference Selection Bits**

These bits select the voltage reference for the ADC, as shown in Table 24-3. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set). The internal voltage reference options may not be used if an external reference voltage is being applied to the AREF pin.

Table 24-3. Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal V_{ref} turned off
0	1	AV_{CC} with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 1.1V Voltage Reference with external capacitor at AREF pin

- **Bit 5 – ADLAR: ADC Left Adjust Result**

The ADLAR bit affects the presentation of the ADC conversion result in the ADC Data Register. Write one to ADLAR to left adjust the result. Otherwise, the result is right adjusted. Changing the ADLAR bit will affect the ADC Data Register immediately, regardless of any ongoing conversions. For a complete description of this bit, see "ADCL and ADCH – The ADC Data Register" on page 250.

- **Bit 4 – Reserved**

This bit is an unused bit in the ATmega48A/PA/88A/PA/168A/PA/328/P, and will always read as zero.

- **Bits 3:0 – MUX[3:0]: Analog Channel Selection Bits**

The value of these bits selects which analog inputs are connected to the ADC. See Table 24-4 for details. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set).

Table 24-4. Input Channel Selections

MUX3...0	Single Ended Input
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5
0110	ADC6
0111	ADC7
1000	ADC8[1]
1001	(reserved)
1010	(reserved)
1011	(reserved)
1100	(reserved)
1101	(reserved)
1110	1.1V (V_{BG})
1111	0V (GND)

Note: 1. For Temperature Sensor.

Figura 19. Control del ADC_channel, referencia de tensión, y orden de lectura de los bytes.

24.9.2 ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ADEN: ADC Enable**

Writing this bit to one enables the ADC. By writing it to zero, the ADC is turned off. Turning the ADC off while a conversion is in progress, will terminate this conversion.

- **Bit 6 – ADSC: ADC Start Conversion**

In Single Conversion mode, write this bit to one to start each conversion. In Free Running mode, write this bit to one to start the first conversion. The first conversion after ADSC has been written after the ADC has been enabled, or if ADSC is written at the same time as the ADC is enabled, will take 25 ADC clock cycles instead of the normal 13. This first conversion performs initialization of the ADC.

ADSC will read as one as long as a conversion is in progress. When the conversion is complete, it returns to zero. Writing zero to this bit has no effect.

- **Bit 5 – ADATE: ADC Auto Trigger Enable**

When this bit is written to one, Auto Triggering of the ADC is enabled. The ADC will start a conversion on a positive edge of the selected trigger signal. The trigger source is selected by setting the ADC Trigger Select bits, ADTS in ADCSRB.

- **Bit 4 – ADIF: ADC Interrupt Flag**

This bit is set when an ADC conversion completes and the Data Registers are updated. The ADC Conversion Complete Interrupt is executed if the ADIE bit and the I-bit in SREG are set. ADIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ADIF is cleared by writing a logical one to the flag. Beware that if doing a Read-Modify-Write on ADCSRA, a pending interrupt can be disabled. This also applies if the SBI and CBI instructions are used.

- **Bit 3 – ADIE: ADC Interrupt Enable**

When this bit is written to one and the I-bit in SREG is set, the ADC Conversion Complete Interrupt is activated.

- **Bits 2:0 – ADPS[2:0]: ADC Prescaler Select Bits**

These bits determine the division factor between the system clock frequency and the input clock to the ADC.

Table 24-5. ADC Prescaler Selections

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

Figura 19. ENABLE del ADC, START CONVERSION y PREESCALER.

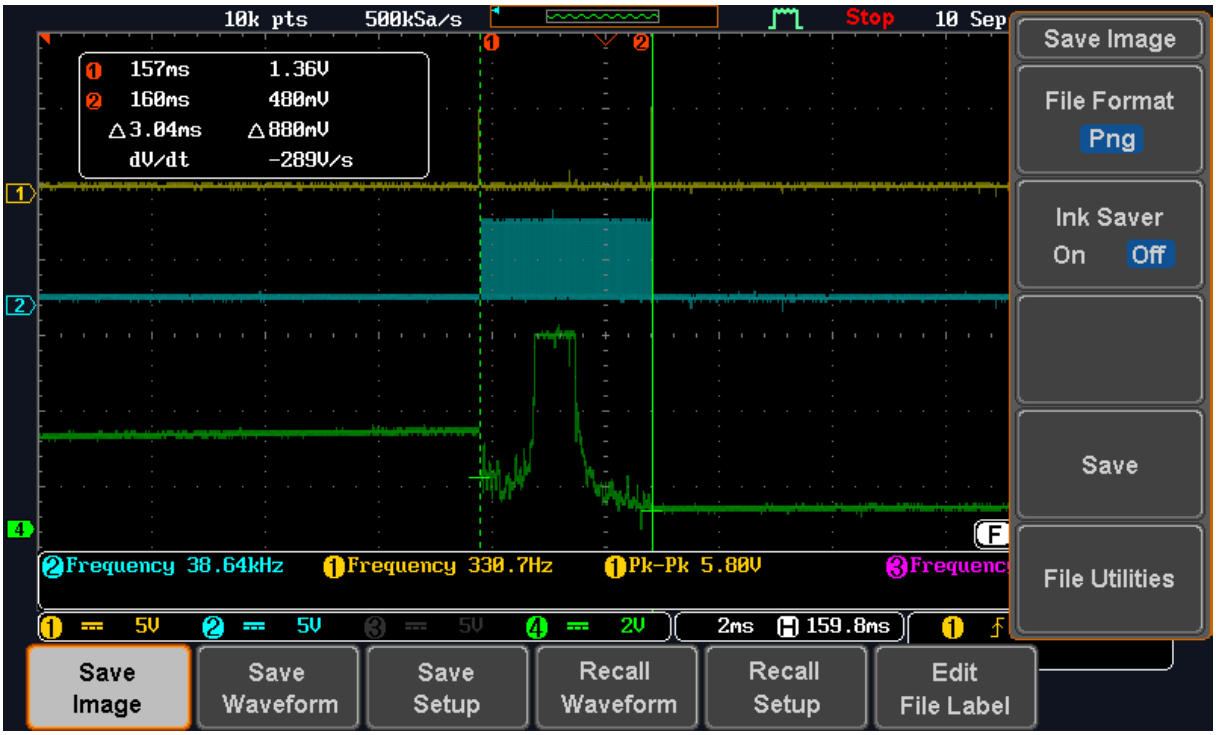


Figura 20. Amarillo = START; Azul = CLK ; Verde = sensor output
Con el ADC activo la señal CLK es algo asimétrica. La conversión del array se ha incrementado un poco $\Delta t_{128PIXELS}$ **≈3ms** frente a 2.25ms.

7.6 Operación con el ADC.

En cada paso de motor se deben leer los 128 pixels del sensor. En cada pulso de CLK se llama al ADC y se almacenan sus valores en un vector para su posterior transmisión con la UART.

```
for(int i=0;i< nPixel;i++)
      {
      Ejecutar CLK_HIGH ;
      Ejecutar CLL_LOW ;

      // Iniciamos conversión
      ADCSRA |= (1 << ADSC); // ADC START CONVERSION
      while (bit_is_set(ADCSRA, ADSC)); // ADSC se borra cuando la conversion finalize
      //
      low = ADCL;
      high = ADCH;
      //
      vectorL[i]= low ;
      vectorH[i]= high ;
      }
```

Figura 21. Código de lectura y almacenamiento de los 128x2 bytes obtenidos por el ADC.

APÉNDICE 1.

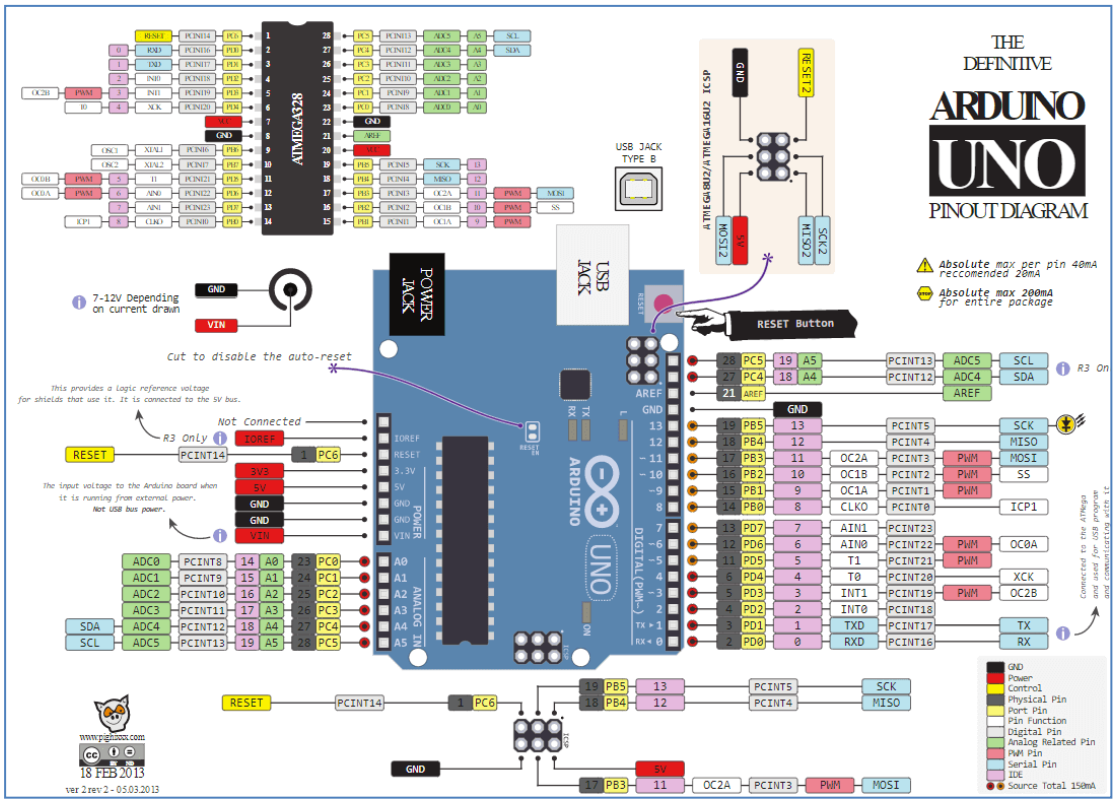
```
void Reading_TSL1401_Sensor()
 {
 uint8_t low, high ;

 // MOVIMIENTO DEL MOTOR SIN PARAR

 /*
 digitalWrite(Pin_Motor,HIGH);
  delay(1);
 digitalWrite(Pin_Motor,LOW) ;
  delay(1) ;
 digitalWrite(Pin_Motor,HIGH);
  delay(1);
 digitalWrite(Pin_Motor,LOW) ;
  delay(1) ;
  */

 // START SEQUENCE
 digitalWrite(TSL1401_ST,LOW); digitalWrite(TSL1401_CLK,LOW);  delayMicroseconds(2);
 digitalWrite(TSL1401_ST,HIGH); digitalWrite(TSL1401_CLK,LOW); delayMicroseconds(2);
 digitalWrite(TSL1401_ST,HIGH); digitalWrite(TSL1401_CLK,HIGH); delayMicroseconds(2);
 digitalWrite(TSL1401_ST,LOW); digitalWrite(TSL1401_CLK,HIGH); delayMicroseconds(2);
 digitalWrite(TSL1401_ST,LOW); digitalWrite(TSL1401_CLK,LOW);  delayMicroseconds(2);
 //
 for(int i=0;i< nPixel;i++)
    {
    digitalWrite(TSL1401_CLK,HIGH);  delayMicroseconds(5);
    digitalWrite(TSL1401_CLK,LOW);  delayMicroseconds(5);
    // Iniciamos conversion
    /*
    ADCSRA |= (1 << ADSC); // ADC START CONVERSION
    //sbi(ADCSRA, ADSC);
    while (bit_is_set(ADCSRA, ADSC)); // ADSC se borra cuando la conversion finaliza
    low = ADCL;
    high = ADCH;
    vectorL[i]= low ;
    vectorH[i]= high ;
    vector[i] = (high << 8) | low  ; // combinación de los dos bytes sdel ADC
    */
    }
 // SE REPITE LA SECUENCIA DE START
 // CUANDO SE TRANSMITE EN SERIE EL TIEMPO DE INTEGRACION SIGUE TRABAJANDO
 digitalWrite(TSL1401_ST,LOW); digitalWrite(TSL1401_CLK,LOW);  delayMicroseconds(2);
 digitalWrite(TSL1401_ST,HIGH); digitalWrite(TSL1401_CLK,LOW); delayMicroseconds(2);
 digitalWrite(TSL1401_ST,HIGH); digitalWrite(TSL1401_CLK,HIGH); delayMicroseconds(2);
 digitalWrite(TSL1401_ST,LOW); digitalWrite(TSL1401_CLK,HIGH); delayMicroseconds(2);
 digitalWrite(TSL1401_ST,LOW); digitalWrite(TSL1401_CLK,LOW);  delayMicroseconds(2);

}
////////////////////////////////////////////////////////////////////

void loop()
 {
 Reading_TSL1401_Sensor() ;
 delay(1) ;
 } // END LOOP
```

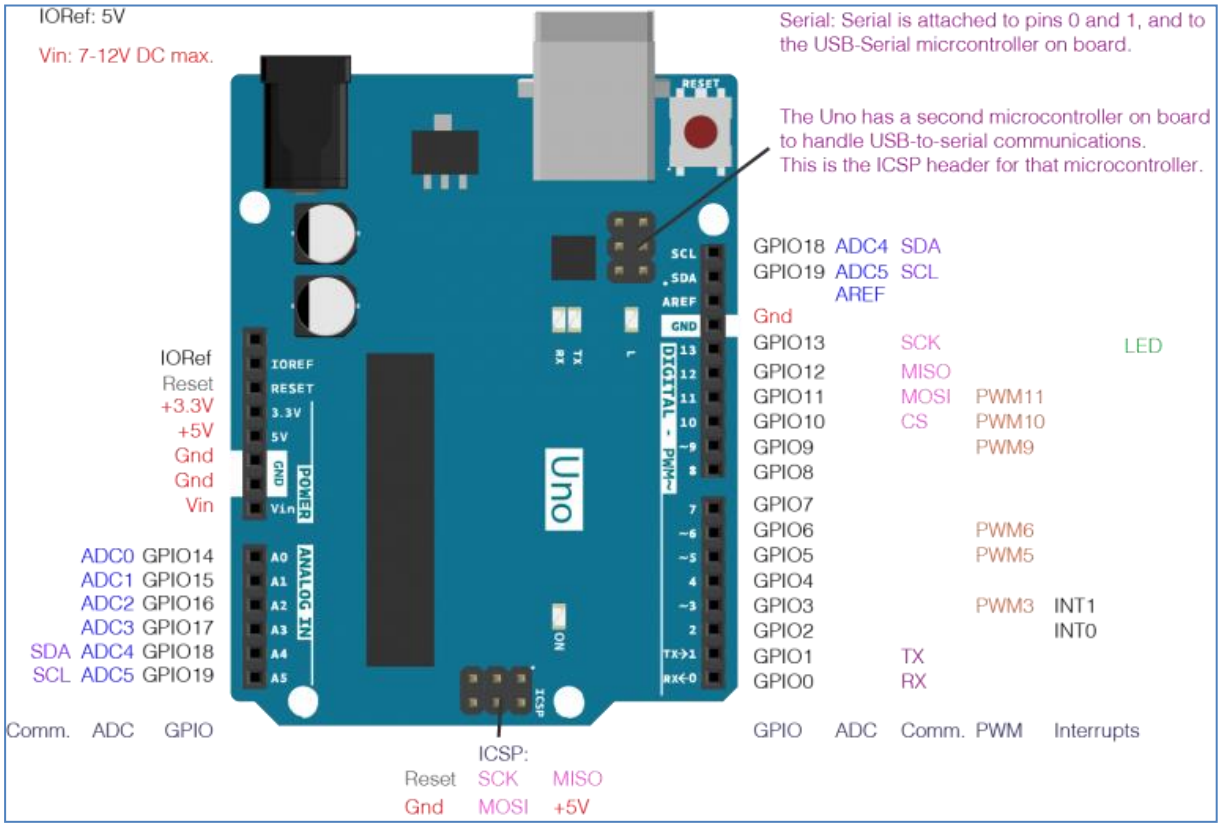
APÉNDICE 2.

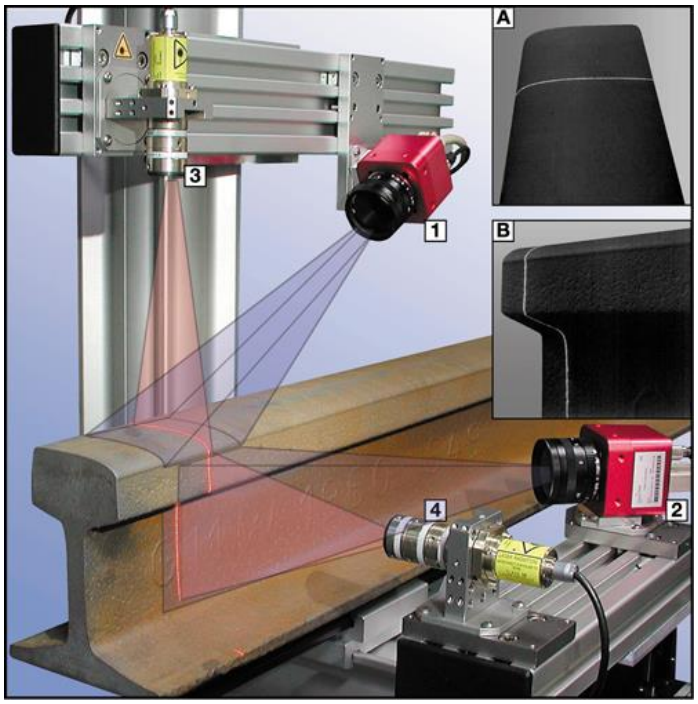
```
void ADC_CONFIG()
  {
  // Borramos el registro ADCSRA y registro ADCSRB
  ADCSRA = 0; // ADC Control and Status Register A
  ADCSRB = 0; // ADC Control and Status Register B
  //
  // ADMUX - ADC Multiplexer Selection Register.
  ADMUX |= (0 & 0x00);    // SELECCIONAMOS CHANNEL 0
  //Configuramos voltage de referencia
  ADMUX |= (1 << REFS0);  // REFERENCE = VCC
  //ADMUX |= (1 << ADLAR);  // LEFT ADJUST
  // cbi(ADMUX,REFS1) ;
  // sbi(ADMUX,REFS0) ;

  //Seleccionamos A0 como un pin de entrada analogico
  // Nota: segun datasheer el tiempo de conversion son 13 ciclos de reloj del ADC
  //ADCSRA |= (1 << ADPS2) | (1 << ADPS0); // 32 prescaler for 38.5 KHz
  //ADCSRA |= (1 << ADPS2); // 16 prescaler for 76.9 KHz
  //ADCSRA |= (1 << ADPS1) | (1 << ADPS0);

  // 8 prescaler for 153.8 KHz ADCSRA |= (1 << ADPS1);
  // 4 prescaler a 307.7 KHz
  // ADCSRA |= (1 << ADPS0); // 2 prescaler a 615.2 DEMASIADO RAPIDO
  // ADCSRA |= (1 << ADATE); // habilita auto trigger
  // ADCSRA |= (1 << ADIE); // habilita interrupcion cuando la medida se ha completado
  // 8 prescaler for 153.8 KHz

  ADCSRA |= (1 << ADPS1) ; // 4 prescaler a 307.7 KHz OK
  // ADCSRA |= (1 << ADPS2) ; // 16 DEMASIADO LENTO

  ADCSRA |= (1 << ADEN); // ADC  ENABLE //
}
```

APÉNDICE 3.

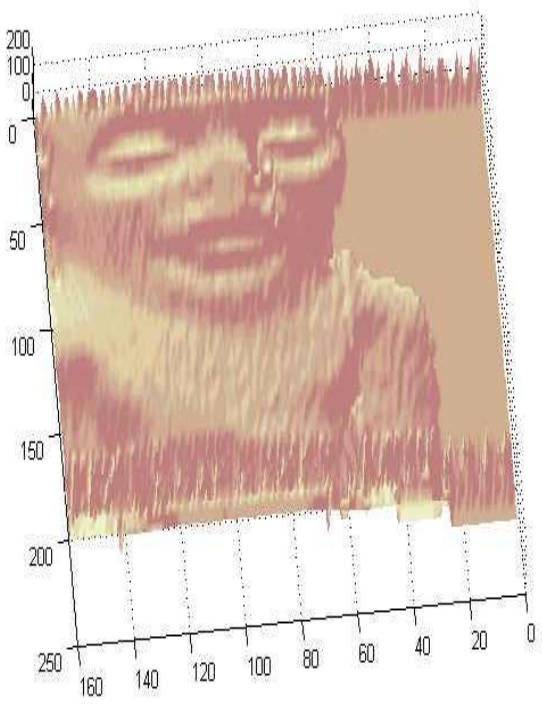
```
   %Leemos un primer byte
   aux1 = fread(arduino,1) ;
   % Leemos un segundo byte
   aux2 = fread(arduino,1)    ;
   if aux1 == 255 && aux2 == 0
         %cabecera_detected = 1 ;
         frame = frame + 1 ;
         disp(sprintf('Head detected %d',frame)) ;
       for pixel=1:128
          byte_bajo = fread(arduino,1) ;    % Primer byte
          byte_alto = fread(arduino,1) ;    % Segundo byte
          if byte_alto >= 4
            disp(sprintf('\nError: frame %d aux1=%d aux2=%d pixel=%d bajo=%d alto=%d , frame, aux1,aux2,pixel,
byte_bajo, byte_alto)) ;
             pause(0.5) ;
             disp('break') ;
             break ;
             end
             Amplitud(pixel) = byte_bajo  + byte_alto*256 ;
       end
%
       clf ;
       plot(Amplitud,'*') ;
       xlabel('#pixel') ;
       ylabel('Intensity') ;
       set(gca, 'FontSize',12, 'FontWeight','bold','YTick',[0 64 128 256 512 768 1024]);
       axis([0 128 0 1024]) ;
       grid minor ;
         %% aJUSTE
%            tipo_ajuste = fittype('gauss1') ;
%            %tipo_ajuste = fittype('poly2') ;
%            vectorx = [1:128] ;
%            aux_ys = Amplitud ;
%            ajuste = fit(vectorx',aux_ys',tipo_ajuste) ;
%            newY = ajuste(vectorx) ; % ESTO ES MAS RÁPIDO
%            hold on ;
%            plot(vectorx, newY,'r-') ;
         [maximo, posicion_pico(paso),] = max(Amplitud) ;
         line([posicion_pico(paso) posicion_pico(paso)],[0 1024]) ;
         titulo = sprintf('Vuelta = %d Paso = %d Pixel = %d',vuelta, paso, posicion_pico(paso) )
         title(titulo) ;
       end
       flushoutput(arduino) ;
       aux1 = aux2 ; % para salir del bucle
```


THE DEFINITIVE
ARDUINO UNO
PINOUT DIAGRAM
ATMEGA328
USB JACK TYPE B
POWER JACK
USB JACK
RESET Button
7-12V Depending on current drawn
Cut to disable the auto-reset
Not Connected
R3 Only
R3 On
ARDUINO UNO
DIGITAL (PWM~)
ANALOG IN
POWER
Absolute max per pin 40mA reccomended 20mA
Absolute max 200mA for entire package
18 FEB 2013
ver 2 rev 2 - 05.03.2013


IORef: 5V
Vin: 7-12V DC max.
Serial: Serial is attached to pins 0 and 1, and to the USB-Serial micrcontroller on board.
The Uno has a second microcontroller on board to handle USB-to-serial communications.
This is the ICSP header for that microcontroller.
IORef
Reset
+3.3V
+5V
Gnd
Gnd
Vin
ADC0 GPIO14
ADC1 GPIO15
ADC2 GPIO16
ADC3 GPIO17
SDA ADC4 GPIO18
SCL ADC5 GPIO19
Comm. ADC GPIO
ICSP:
Reset SCK MISO
Gnd MOSI +5V
GPIO18 ADC4 SDA
GPIO19 ADC5 SCL
AREF
Gnd
GPIO13 SCK LED
GPIO12 MISO
GPIO11 MOSI PWM11
GPIO10 CS PWM10
GPIO9 PWM9
GPIO8
GPIO7
GPIO6 PWM6
GPIO5 PWM5
GPIO4
GPIO3 PWM3 INT1
GPIO2 INT0
GPIO1 TX
GPIO0 RX
GPIO ADC Comm. PWM Interrupts


A
B
1
2
3
4


5 cm
9 cm


200
100
0
0
50
100
150
200
250
160
140
120
100
80
60
40
20
0