Comunicación SERIAL

TECNOLOGÍAS DE LA INFORMACIÓN

ÁREA DESARROLLO DE SOFTWARE MULTIPLATAFORMA

MTRA. Irene Garcia

Equipo #1
GALVAN COVARRUBIAS VICTOR MANUEL
RODRÍGUEZ OSUNA LUIS FERNANDO
SILVAS PUGA MONSERRATH
YESCAS MORENO TANIA EUNISES

San Luis Rio Colorado, Sonora

Junio, 2021

Temas

- Utilidad o funcionamiento tanto en Raspberry como Arduino.
- Cómo establecer la comunicación.
- Librerías en caso de necesitar.
- Ejemplos de proyectos donde se utilice este tipo de comunicación.

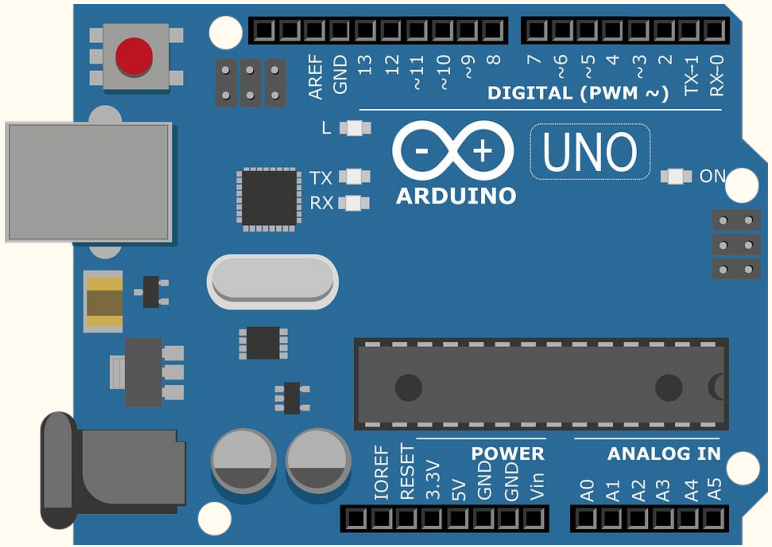
Comunicación en Serie

• La comunicación serie o comunicación secuencial, en telecomunicaciones e informática, es el proceso de envío de datos de un bit a la vez, de forma secuencial, sobre un canal de comunicación o un bus.

La ventaja de la comunicación serie es que necesita un número más pequeño de líneas de transmisión que una comunicación paralela que transmita la misma información.

Comunicación en Serie

•El hardware Arduino tiene soporte incorporado para la comunicación en serie en los pines 0 y 1 (que también va a la computadora a través de la conexión USB). El soporte de serie nativo ocurre a través de una pieza de hardware (integrada en el chip) llamada UART



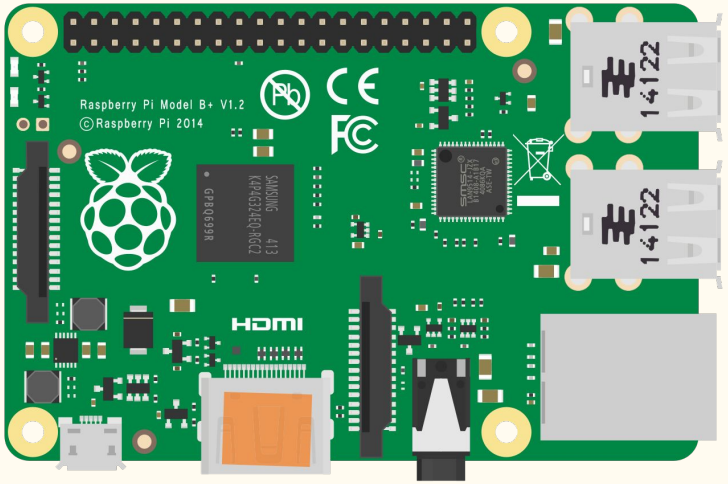
Comunicación en Serie

•La Raspberry Pi contiene un puerto serie UART en el encabezado GPIO en los pines 8, TXD (GPIO 14) y 10, RXD (GPIO 15).

El puerto serie es una forma de bajo nivel de enviar datos entre la Raspberry Pi y otro sistema informático. Hay dos formas principales de utilizarlo:

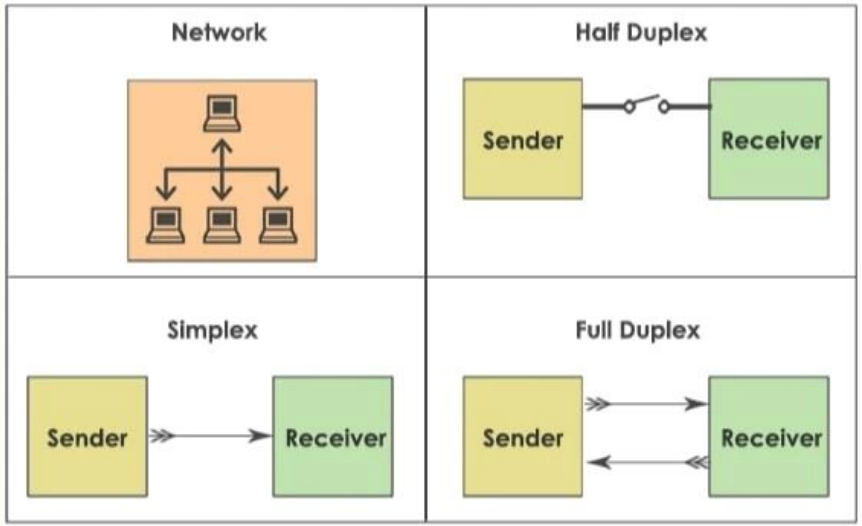
Conectarse a una PC para permitir el acceso a la consola de Linux. Esto puede ayudar a solucionar problemas durante el arranque o iniciar sesión en Raspberry Pi si el video y la red no están disponibles.

Conexión a un microcontrolador u otro periférico que tenga una interfaz en serie. Esto puede resultar útil si desea que la Raspberry Pi controle otro dispositivo.



Cómo establecer la comunicación serial?

- Para implementar la comunicación en serie, se requieren un origen y un destino.
- El método Simplex implementa la transmisión de datos unidireccional.
- Modo Half Duplex permite que el origen y el destino estén activos, pero no simultáneamente.
- Modo Full Duplex es la forma de comunicación serial más utilizada en el mundo.

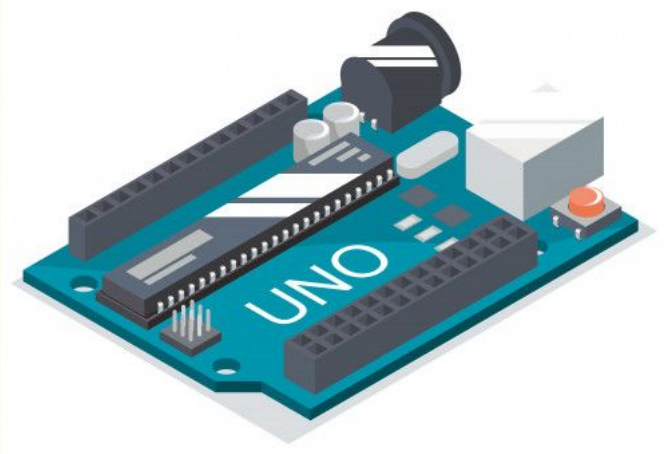


Librerías en caso de necesitar

Arduino:

El puerto serie de Arduino son la forma principal de comunicar una placa Arduino con un ordenador.
Para realizar la conexión mediante puerto serie únicamente es necesario conectar nuestra placa empleando el mismo puerto que empleamos para programarlo.

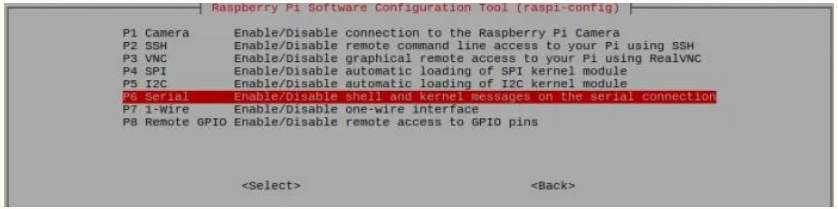
No es necesario descargar alguna librería.



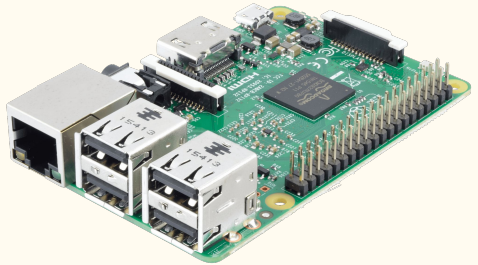
Librerías en caso de necesitar

Raspberry:

Para activar la comunicación serial debemos habilitar el hardware. Puede hacerse de manera sencilla,con raspi-config. Ejecutamos el comando `sudo raspi-config`, luego vamos a Interfacing options, seleccionamos P6 Serial y confirmamos.



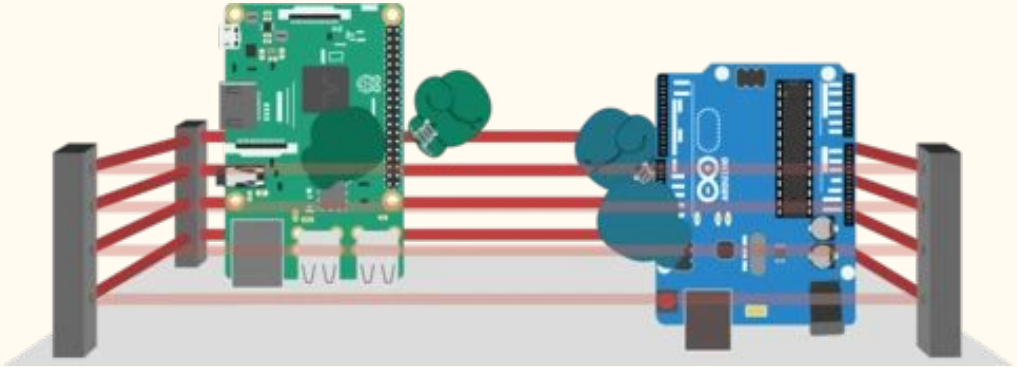
Al final solo hay que reiniciar la Raspberry para que la configuración se guarde correctamente



Librerías en caso de necesitar

Para la conexión de puertos entre Raspberry y Arduino a través del puerto serie se necesita instalar la librería *pyhton-serial* con el comando **`apt install python-serial`**, esta se coloca en la terminal de la Raspberry para poder lograr la comunicación con Arduino.

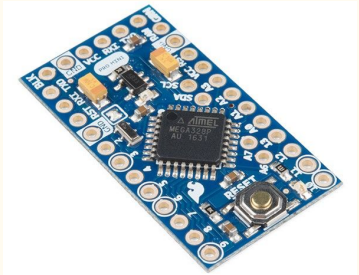
Gracias a esta librería, podríamos decir que Arduino sería el esclavo de Raspberry, ya que las ventajas que nos ofrece Arduino para controlar diferentes sensores y actuadores, haciendo que se comuniquen entre ellos a través del puerto serie.

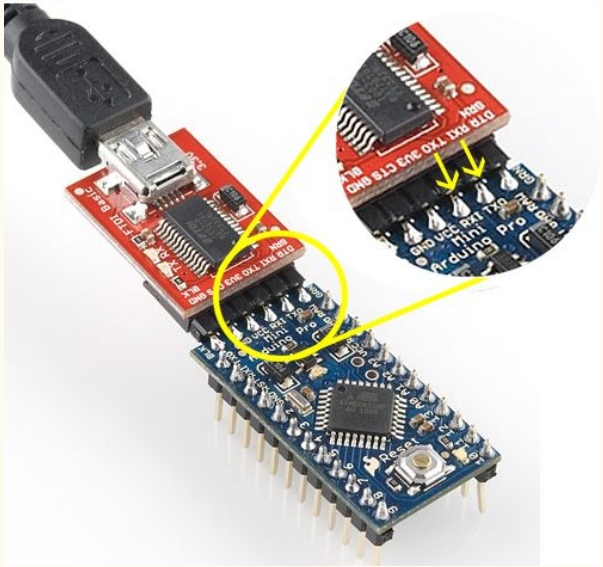


Ejemplos de proyectos donde se utilice este tipo de comunicación.

Para adquisición de datos, control, depuración de código, etc.

Sistemas embebidos



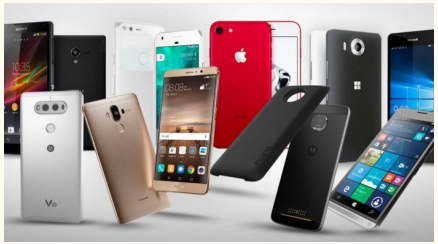


Sistemas embebidos independientes.

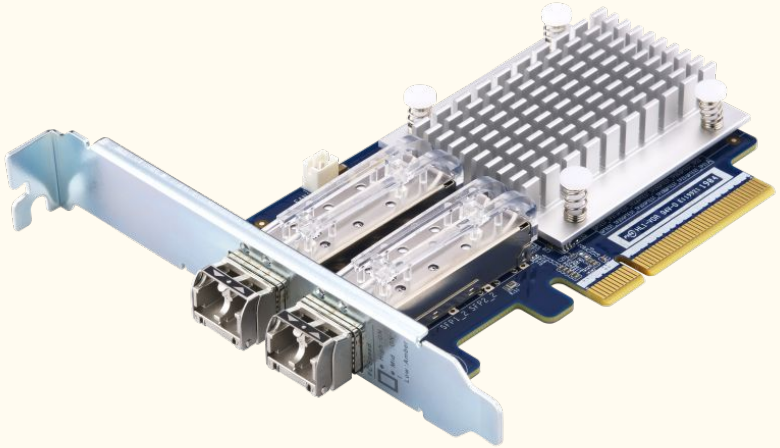
Sistemas embebidos en tiempo real.

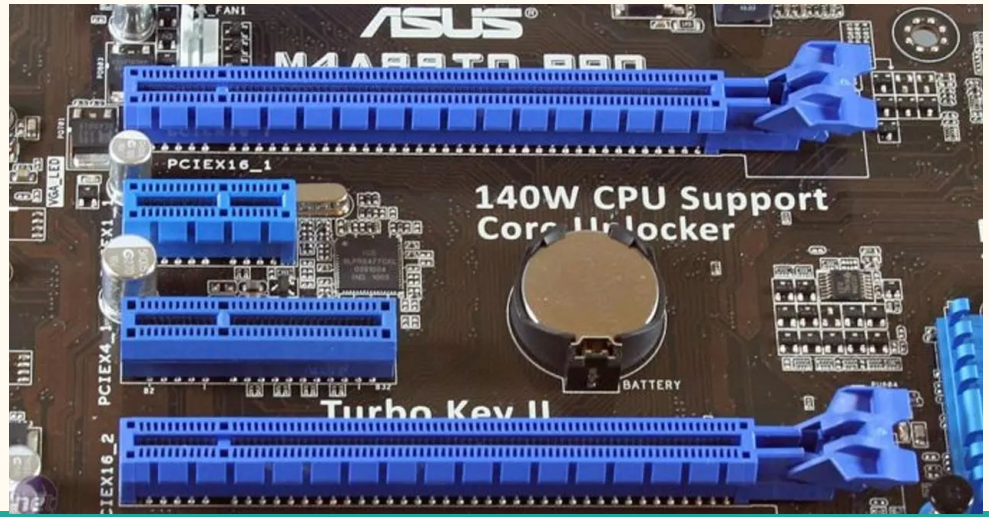
Sistemas embebidos en red.

Sistemas embebidos móviles.

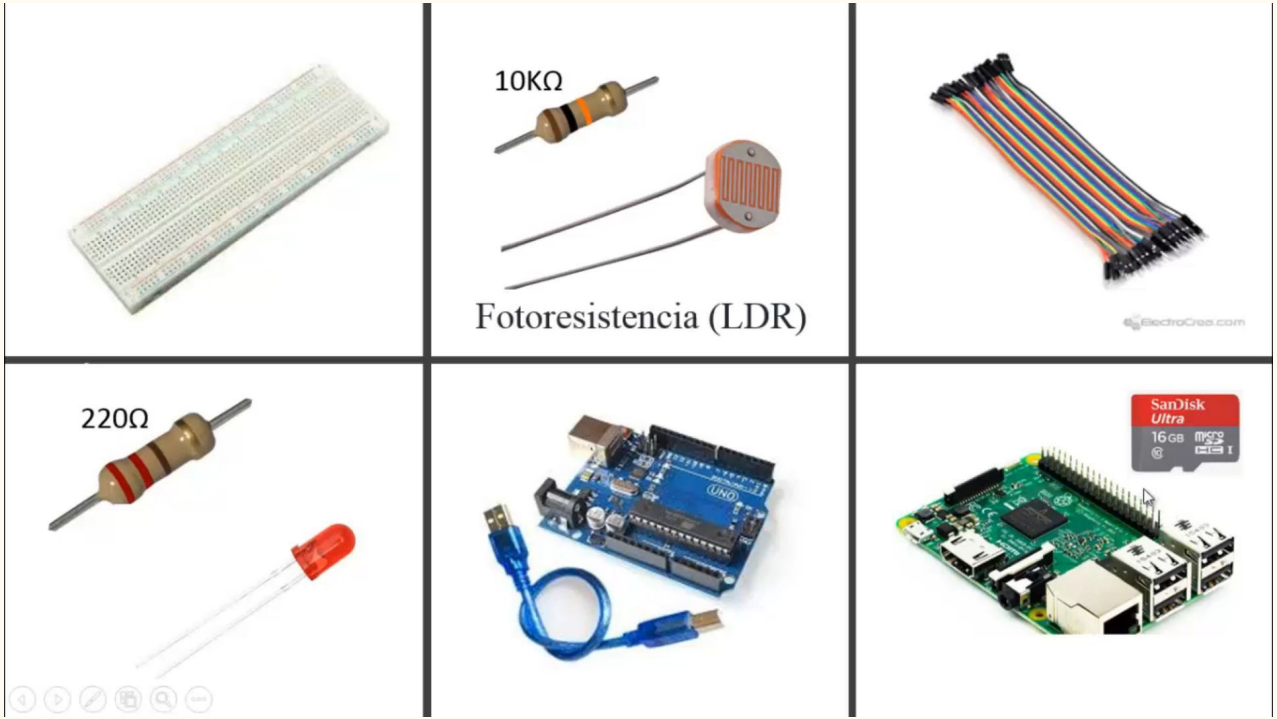


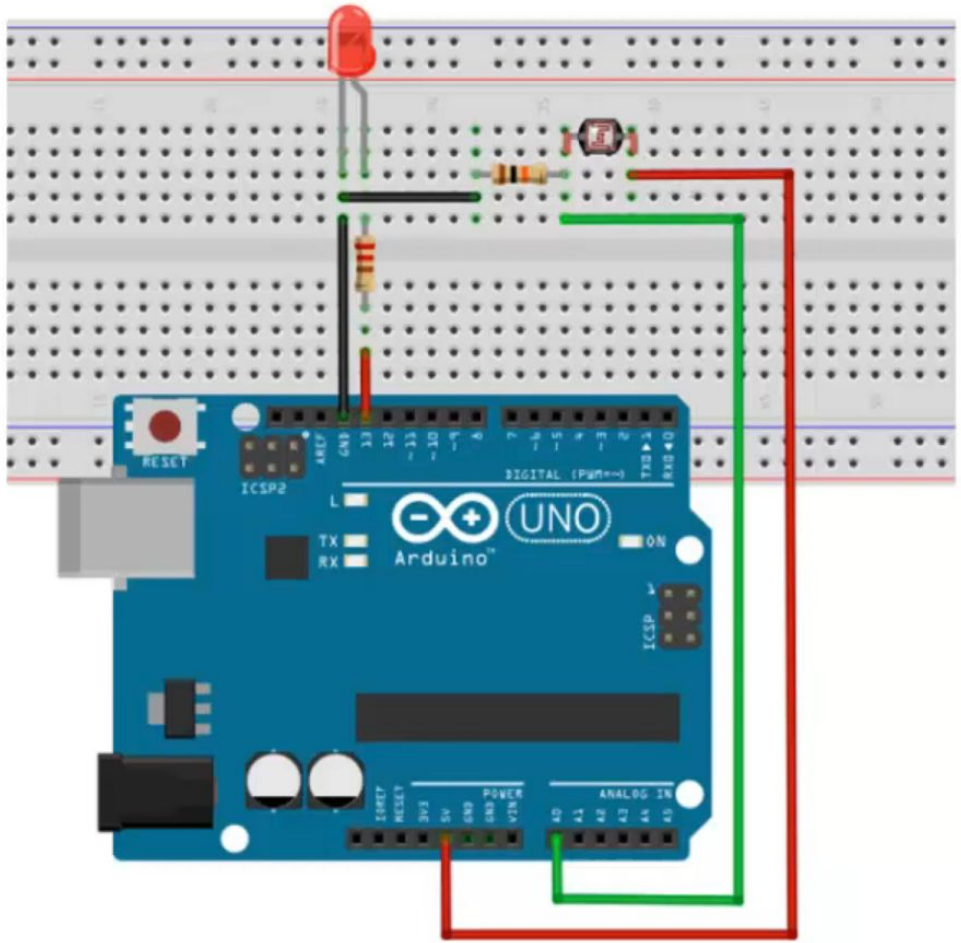

www.med




ASUS
FAN1
PCIEX16_1
140W CPU Support
BATTERY
Turbo Key II

Práctica




RESET
ICSP2
UNO
Arduino
TX
RX
ON
ICSP
POWER
ANALOG IN

PracticaSerial Arduino 1.8.15 (Windows Store 1.8.49.0)

Archivo Editar Programa Herramientas Ayuda

PracticaSerial

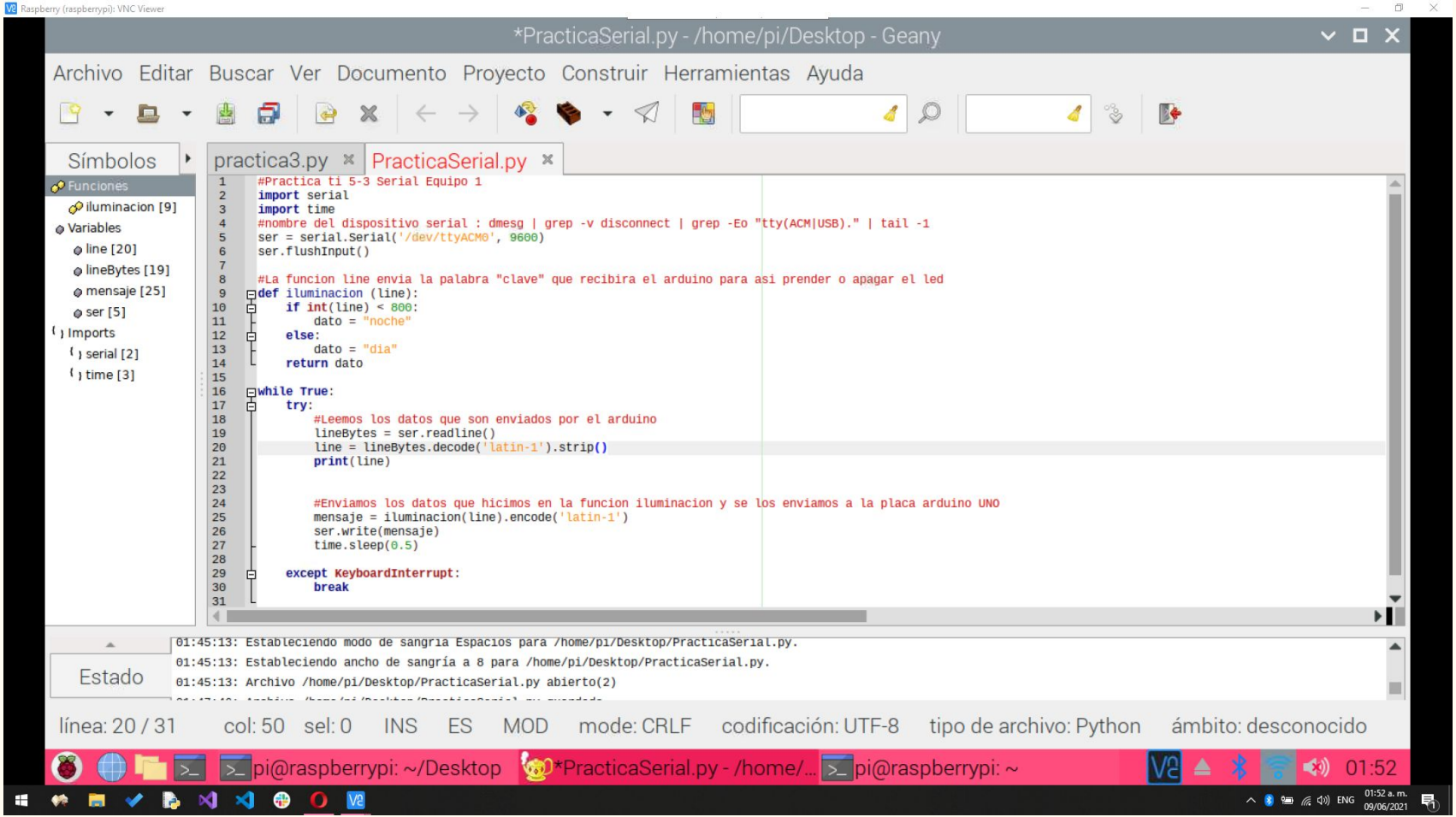
```
//TI 5-3 Practica Serial Equipo 1
//Declaramos nuestros elementos
int pinAnalogico = A0;
int led = 13;
void setup() {
  //establecemos la resistencia como entrada y el led como salida
  pinMode (pinAnalogico,INPUT);
  pinMode (led, OUTPUT);
  //Inicialisamos la comunicacion serial
  Serial.begin(9600);
}
void loop() {
  //leemos la resistencia y se la asignamos a una variable
  int valorPinAnalogico = analogRead(pinAnalogico);
  //Imprimimos el valor de la resistecia
  Serial.println(valorPinAnalogico);
  delay(100);

  //Si la comunicacion serial se encuentra disponible empezaomos a recibir la informacion
  if (Serial.available()>0){
    String dato = Serial.readStringUntil('\n');
    if (dato == "noche"){
      //Si el raspberry nos envia la palabra "noche" encendemos el led
      digitalWrite(led, HIGH);
    }else if (dato == "dia"){
      //Si el raspberry nos envia la palabra "dia" apagamos el led
      digitalWrite(led, LOW);
      }
    }delay(100);
}
```

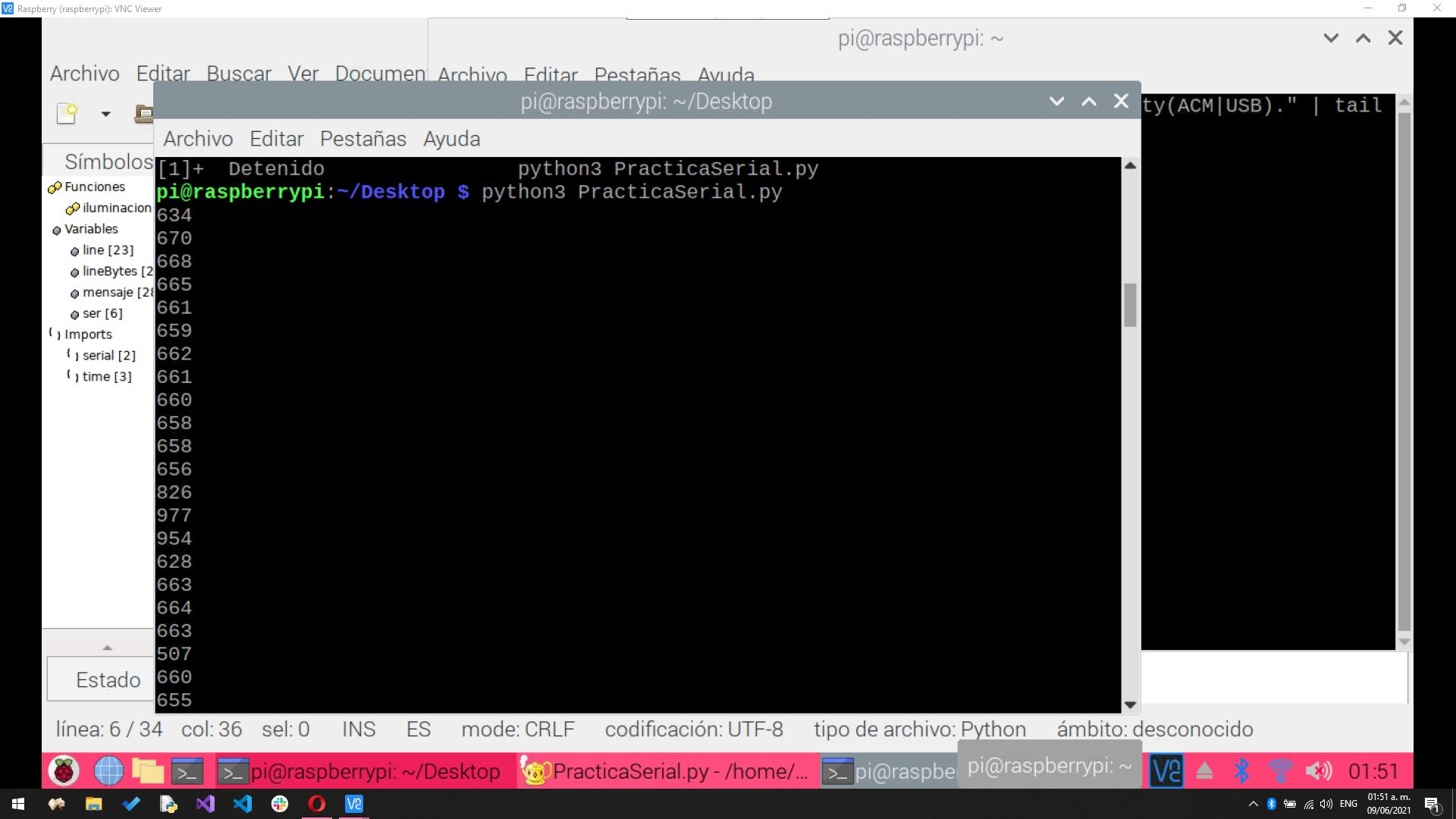
Subido

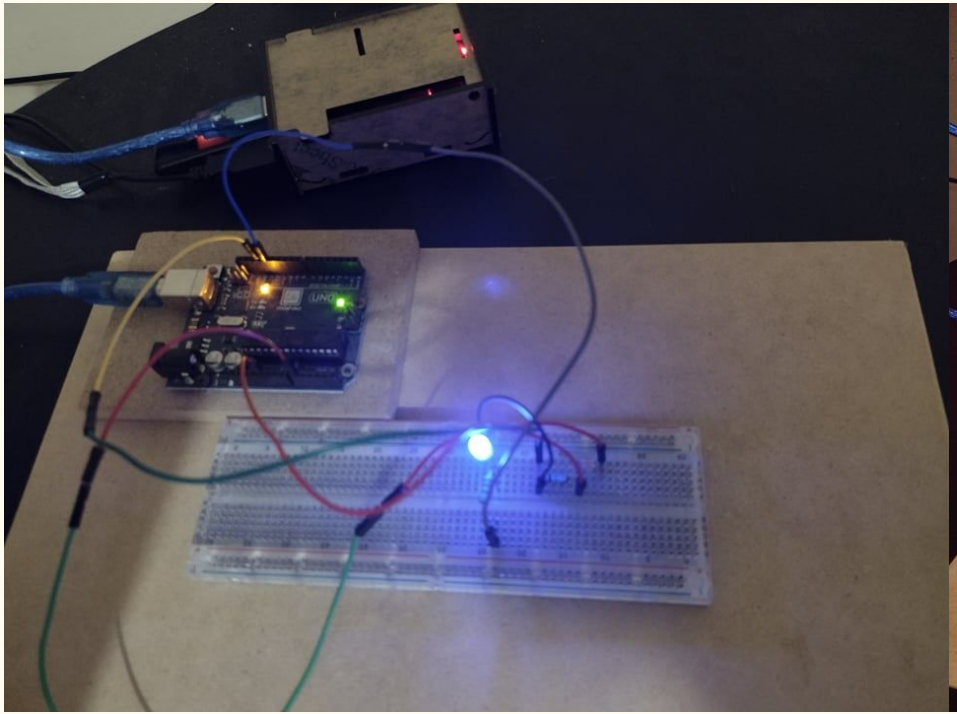
```
Las variables Globales usan 208 bytes (10%) de la memoria dinámica, dejando 1840 bytes para las variables locales. El máximo es 2048 bytes.
```

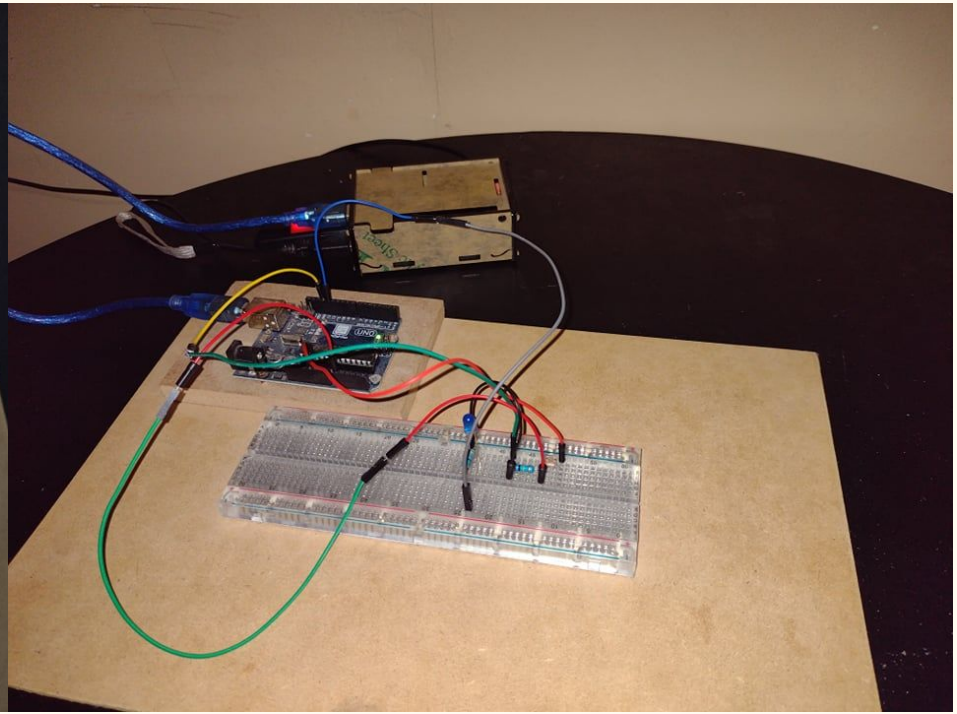
30 Arduino Uno en COM5

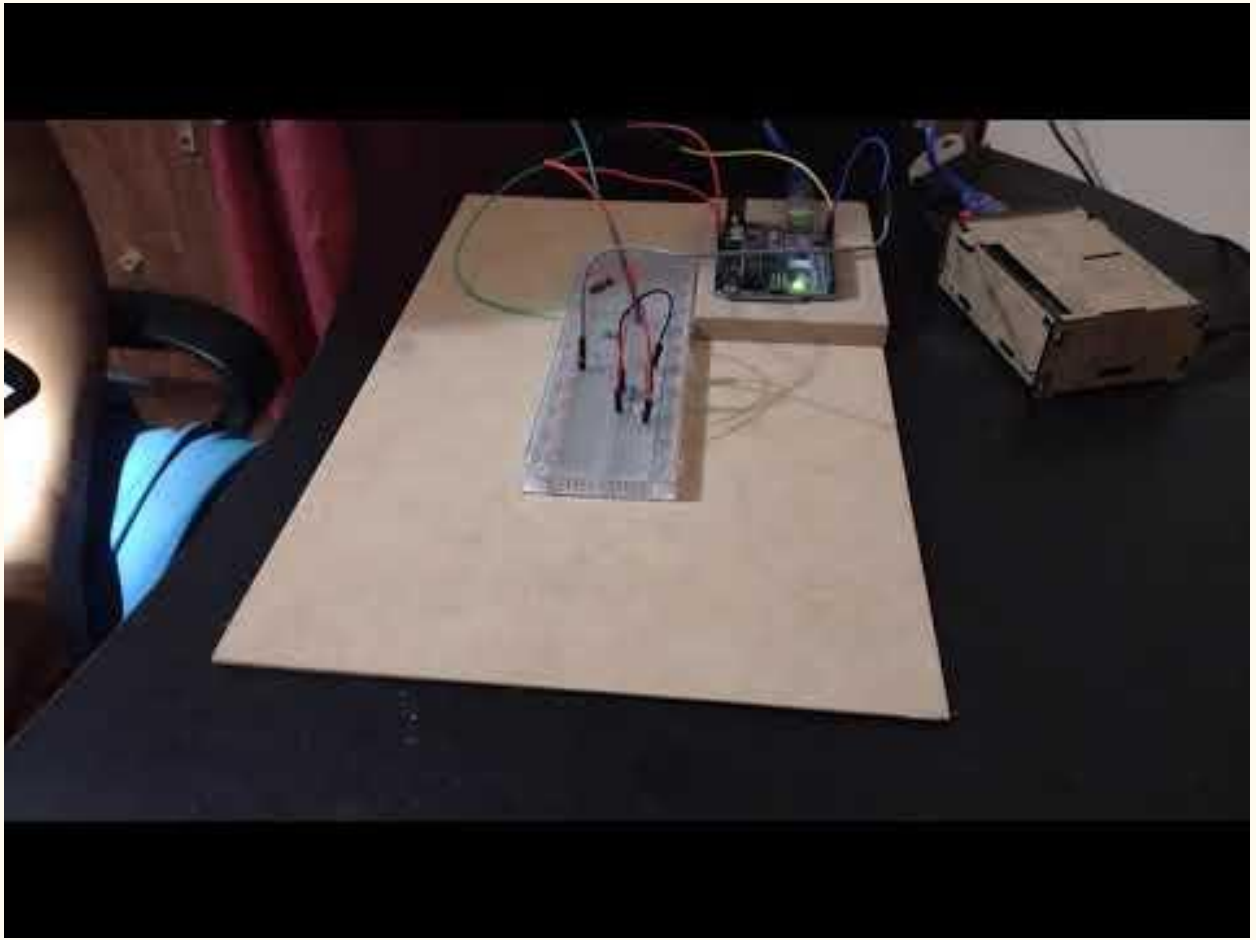
ENG 01:30 a. m. 09/06/2021


Raspberry (raspberrypi): VNC Viewer
*PracticaSerial.py - /home/pi/Desktop - Geany
Archivo Editar Buscar Ver Documento Proyecto Construir Herramientas Ayuda
Símbolos
Funciones
iluminacion [9]
Variables
line [20]
lineBytes [19]
mensaje [25]
ser [5]
Imports
serial [2]
time [3]
practica3.py
PracticaSerial.py
#Practica ti 5-3 Serial Equipo 1
import serial
import time
#nombre del dispositivo serial : dmesg | grep -v disconnect | grep -Eo "tty(ACM|USB)." | tail -1
ser = serial.Serial('/dev/ttyACM0', 9600)
ser.flushInput()
#La funcion line envia la palabra "clave" que recibira el arduino para asi prender o apagar el led
def iluminacion (line):
if int(line) < 800:
dato = "noche"
else:
dato = "dia"
return dato
while True:
try:
#Leemos los datos que son enviados por el arduino
lineBytes = ser.readline()
line = lineBytes.decode('latin-1').strip()
print(line)
#Enviamos los datos que hicimos en la funcion iluminacion y se los enviamos a la placa arduino UNO
mensaje = iluminacion(line).encode('latin-1')
ser.write(mensaje)
time.sleep(0.5)
except KeyboardInterrupt:
break
Estado
01:45:13: Estableciendo modo de sangría Espacios para /home/pi/Desktop/PracticaSerial.py.
01:45:13: Estableciendo ancho de sangría a 8 para /home/pi/Desktop/PracticaSerial.py.
01:45:13: Archivo /home/pi/Desktop/PracticaSerial.py abierto(2)
línea: 20 / 31 col: 50 sel: 0 INS ES MOD mode: CRLF codificación: UTF-8 tipo de archivo: Python ámbito: desconocido
pi@raspberrypi: ~/Desktop
*PracticaSerial.py - /home/...
pi@raspberrypi: ~
01:52
ENG
01:52 a. m.
09/06/2021


Raspberry (raspberrypi): VNC Viewer
pi@raspberrypi: ~
Archivo Editar Buscar Ver Documen
Archivo Editar Pestañas Ayuda
pi@raspberrypi: ~/Desktop
Archivo Editar Pestañas Ayuda
[1]+ Detenido python3 PracticaSerial.py
pi@raspberrypi:~/Desktop $ python3 PracticaSerial.py
634
670
668
665
661
659
662
661
660
658
658
656
826
977
954
628
663
664
663
507
660
655
ty(ACM|USB)." | tail
Símbolos
Funciones
iluminacion
Variables
line [23]
lineBytes [2
mensaje [28
ser [6]
Imports
serial [2]
time [3]
Estado
línea: 6 / 34 col: 36 sel: 0 INS ES mode: CRLF codificación: UTF-8 tipo de archivo: Python ámbito: desconocido
pi@raspberrypi: ~/Desktop
PracticaSerial.py - /home/...
pi@raspberrypi: ~
01:51
ENG
01:51 a. m.
09/06/2021







Referencias de la práctica

- https://www.youtube.com/watch?v=Z-KoMbs7zsY

- https://github.com/programatumicro/Curso-de-IoT/tree/master/Módulo%207

@programatumicro 

Comunicación SERIAL Equipo #1

Gracias por su atencion.