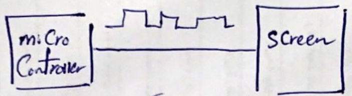
CSE211s:Introduction to Embedded Systems

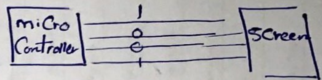
Lect. #6: Communication Systems

Ahmed M. Zaki

"Serial"



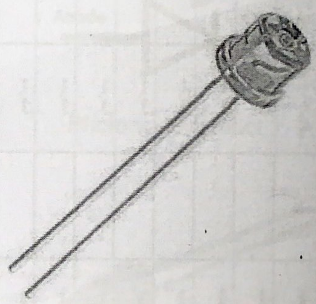
1001 1101



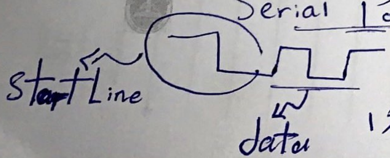
"Parellel"

IR Communication (IR LED)

- An Infrared light-emitting diode (IR LED) is a special purpose LED emitting infrared rays ranging from 700 nm to 1 mm wavelength.
- Different IR LEDs may produce infrared light of differing wavelengths, just like different LEDs produce light of different colors.
- The appearance of IR LED is same as a common LED. Since the human eye cannot see the infrared radiations, it is not possible for a person to identify if an IR LED is working.
- A camera on a cell phone camera solves this problem. The IR rays from the IR LED in the circuit are shown in the camera.

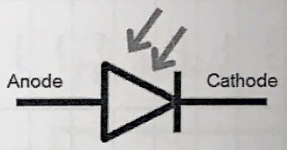


Serial Port

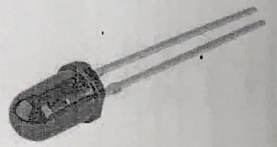


IR Communication
(IR Sensor/ Photodiode)

- An IR sensor is an electronic device that detects IR radiation falling on it. Proximity sensors (used in touchscreen phones and edge avoiding robots), contrast sensors (used in line following robots) and obstruction counters/sensors (used for counting goods and in burglar alarms) are some applications involving IR sensors.

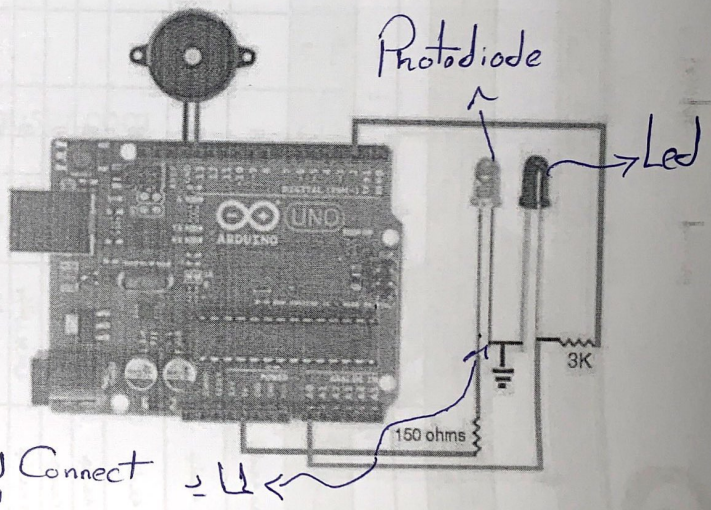


Photodiode symbol



IR Communication

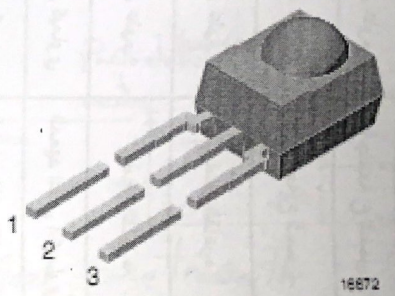
- IR LED and Photodiode can be used to detect obstacles



لازم تحط Resistance و Connect

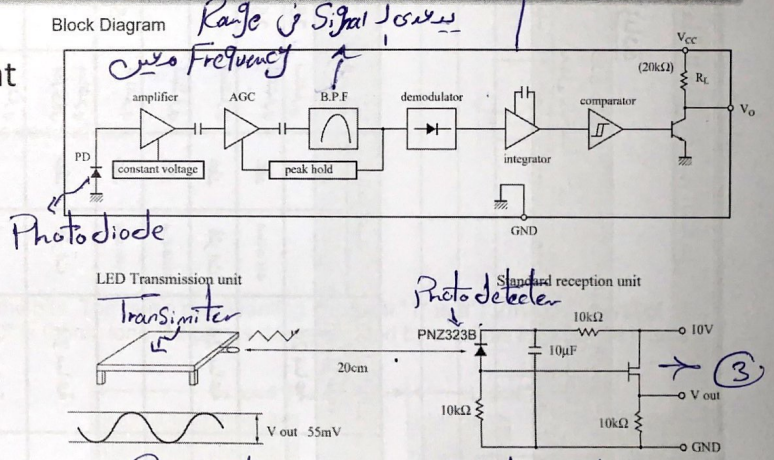
IR Remote controller (IR Receiver)

- Converts IR signal to digital pulses
- Models
  - TSOP4838 and TSOP2438 from www.mouser.com
  - PNA4602 by Panasonic



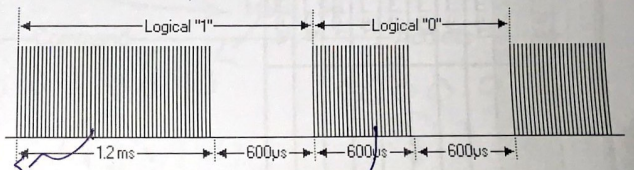
IR Remote controller
(Panasonic Transmitter Specification)

- The light output of the LED transmission unit is adjusted so that the transmission output (V out) of the standard reception unit will be 55 mV when the transmission waveform **(duty = 50%)** is output from the LED transmission unit.
- Here, infrared sensitivity (SIR) of PNZ323B is 0.53 µA when emission illuminance (H) is 12.45 µW/ cm2.



IR Remote controller
Sony SIRC Protocol

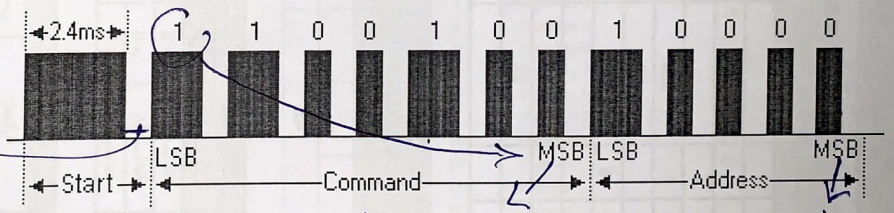
- **Features**
  - 12-bit version, 7 command bits, 5 address bits.
  - 15-bit version, 7 command bits, 8 address bits.
  - 20-bit version, 7 command bits, 5 address bits, 8 extended bits.
  - Pulse width modulation.
  - Carrier frequency of 40kHz.
  - Bit time of 1.2ms or 0.6ms
- **Modulation**
  - The SIRC protocol uses pulse width encoding of the bits. The pulse representing a logical "1" is a 1.2ms long burst of the 40kHz carrier, while the burst width for a logical "0" is 0.6ms long. All bursts are separated by a 0.6ms long space interval. The recommended carrier duty-cycle is 1/4 or 1/3.



IR Remote controller

Sony SIRC Protocol- (cont.)

- **Protocol**
  - The following picture shows a typical pulse train of the 12-bit SIRC protocol. With this protocol the LSB is transmitted first. The start burst is always 2.4ms wide, followed by a standard space of 0.6ms. Apart from signaling the start of a SIRC message this start burst is also used to adjust the gain of the IR receiver. Then the 7-bit Command is transmitted, followed by the 5-bit Device address.
  - In this case Address 1 and Command 19 is transmitted.
  - Commands are repeated every 45ms(measured from start to start) for as long as the key on the remote control is held down.



Time

0 0 1 0 0 1 1

0 0 0 0 1

19

1 address

IR Remote controller
Sony SIRC Protocol- (cont.)

Example Commands

The table below lists some messages sent by Sony remote controls in the 12-bit protocol.

Address	Device
1	TV
2	VCR 1
3	VCR 2
6	Laser Disk
12	Surround Sound
16	Cassette Deck / Tuner
17	CD Player
18	Equalizer

Command	Function
0	Digit key 0
1	Digit key 1
2	Digit key 2
3	Digit key 3
4	Digit key 4
5	Digit key 5
6	Digit key 6
7	Digit key 7
8	Digit key 8
9	Digit key 9
16	Channel +
17	Channel -
18	Volume +
19	Volume -
20	Mute
21	Power

Command	Function
22	Reset
23	Audio Mode
24	Contrast +
25	Contrast -
26	Color +
27	Color -
30	Brightness +
31	Brightness -
38	Balance Left
39	Balance Right
47	Standby

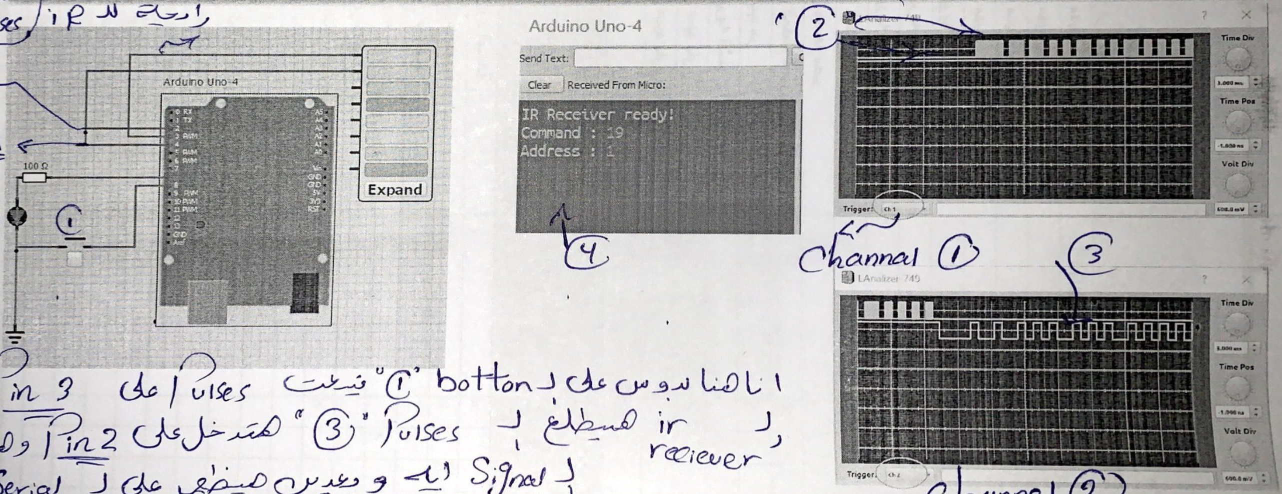
IR Send/Receive Example

لو كنت وصلت الـ Pin 3 بـ Pin 2 direct مكنش هيشتغل لانه الـ Signal اللي هتكون Pulses وانا عايزها stable

راجعة للـ ir

Pin 2 i/P من الـ ir reciever

Pin 4 o/P



Channal ①   ③   Channel ②

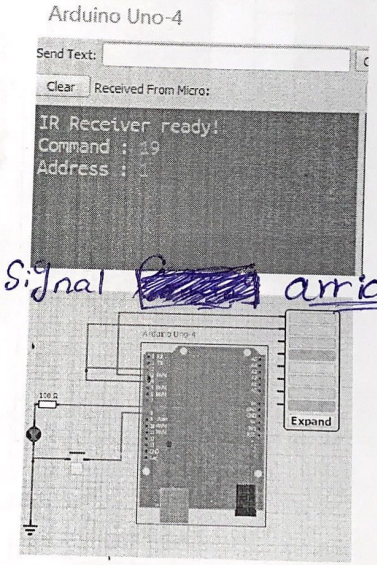
انا هنا ندوس على الـ botton "①" فيطلع Pulses على Pin 3 "②" لـ ir reciever هيطلع الـ Pulses "③" هتدخل على Pin 2 والمنهم نوع الـ Signal ايه وبعدين هيظهر على الـ Serial Port "④" وبعدين الـ Led ⑤ هتنور

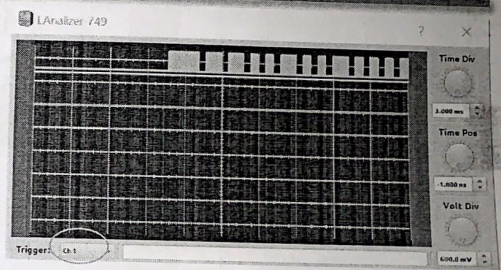
" انا هنا دعمل الـ Simulation الـ Signal اللي جاية من "ir reciever" "

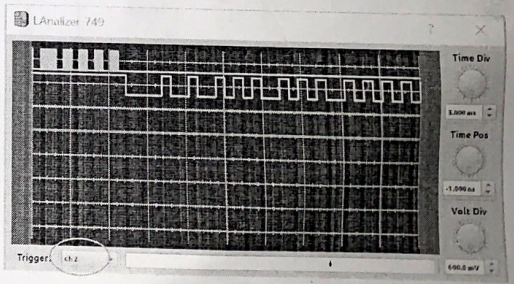
IR Send/Receive Example (cont.)

```
#define IR_SEND_PIN    3
#define IR_RECEIVE_PIN 2
#define LED_PIN 7
#include <IRremote.h>
#define SIM_IR_OUT 4
#define SW_INP 8
IRrecv receiver(IR_RECEIVE_PIN);
IRsend sender;
void setup(){
  pinMode(SW_INP, INPUT_PULLUP);
  pinMode(LED_PIN, OUTPUT);
  pinMode(SIM_IR_OUT, OUTPUT);
  Serial.begin(9600);
  receiver.enableIRIn();
  Serial.println("IR Receiver ready!");
}
```

→ wait until signal [redacted] arrive



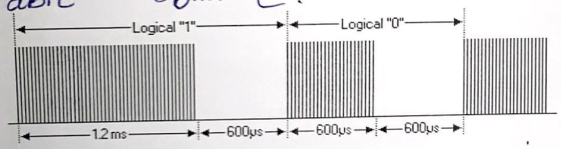


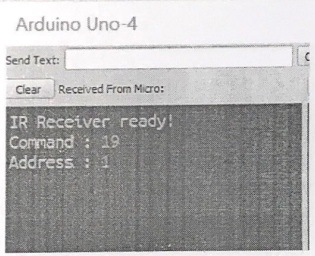


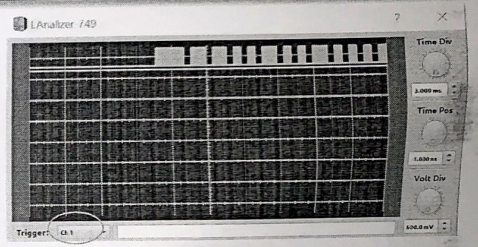
IR Send/Receive Example (cont.)

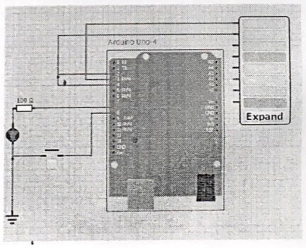
```
17 bool state=0; bool old_state=0;
18 void sendIR(bool v){
19       digitalWrite(SIM_IR_OUT,0);
20       delayMicroseconds((v)?1200:600);
21       digitalWrite(SIM_IR_OUT,1);
22       delayMicroseconds(600);
23 }
```

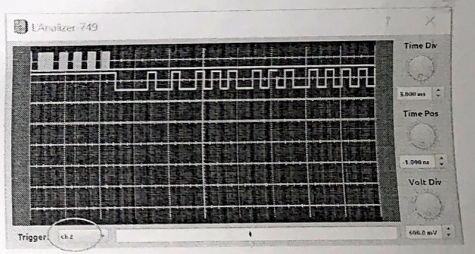
Stable علشان نطلع Signal







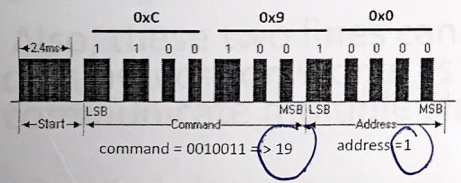


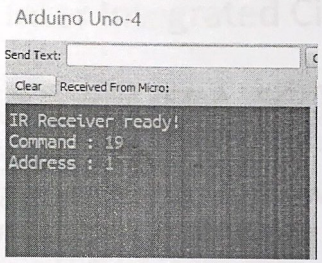


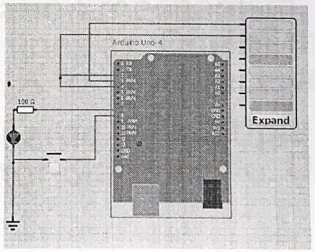
IR Send/Receive Example (cont.)

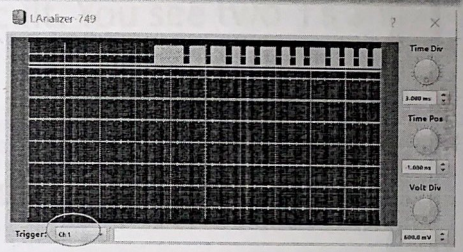
```
24 void loop() {
25 state=digitalRead(SW_INP);
26    if (state && !old_state ){
27       sender.sendSony(0xC90,12);
28       // for simulation only
29       digitalWrite(SIM_IR_OUT,0); delayMicroseconds(2400);
30       digitalWrite(SIM_IR_OUT,1); delayMicroseconds(600);
31       sendIR(1);   sendIR(1);      sendIR(0);      sendIR(0); //0xC
32       sendIR(1);   sendIR(0);      sendIR(0);      sendIR(1); //0x9
33       sendIR(0);   sendIR(0);      sendIR(0);      sendIR(0); //0x0
34    }
35   old_state=state;
36  // decode returns 1 if something was received
37  // otherwise it returns 0
38  if (receiver.decode())
39  {
40   if(receiver.decodedIRData.command == 19)
41         digitalWrite(LED_PIN, !digitalRead(LED_PIN));
42   Serial.print("Command : ");
43   Serial.println(receiver.decodedIRData.command);
44   Serial.print("Address : ");
45   Serial.println(receiver.decodedIRData.address, HEX);
46   receiver.resume(); // Receive the next value
47  }
48 }
```

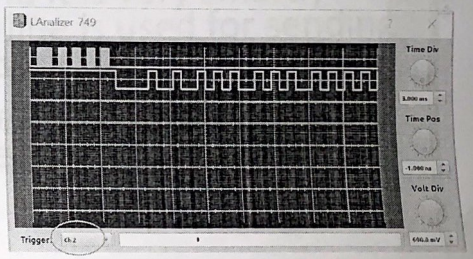
19 (1) وكذلك











(I^2C) Inter Integrated Circuit

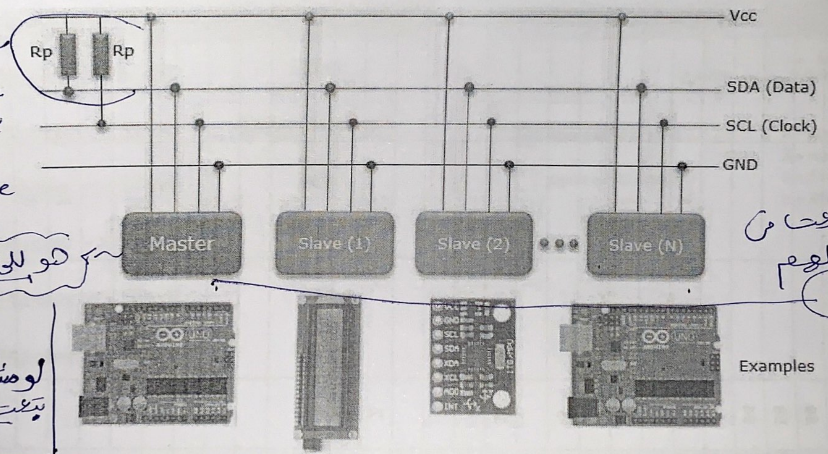
wired Protocol

- I2C or I^2C came from the term Inter Integrated Circuit. You see two 'I's and one 'C'
- I2C is implemented by only Two Wires (SDA, SCL), it is also called TWI (Two Wire Interface). rate Cable , Data , Clock cable "Synchronize"
- It is a kind of serial communication technology which is originally designed for exchanging data between multiple IC chips.
- This is very simple communication mechanism, and It is very smartly designed. It requires only two lines as. One of the line is used for sending Clock signal and the other line is to send/receive data.
- Also, these two lines can be shared by multiple devices. This communication works as Master-Slave mode. One Master can control / communicate multiple Slaves.

(I^2C) Inter Integrated Circuit (Cont.)

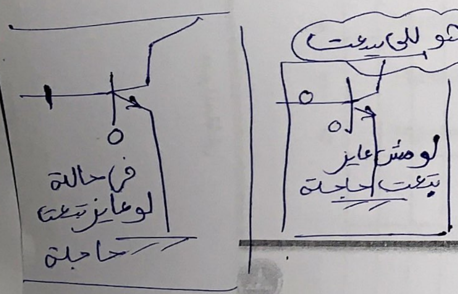
Pull-up Resistor

علشان ال Signals دي معظم
يبقى "open collector" علشان لو
device بعتت ميحرقش التاني



هو اللي يبعت

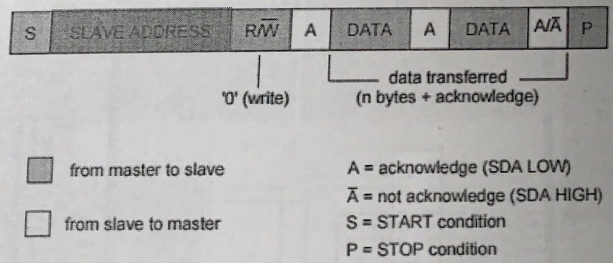
مينفعش كذا واحد يبعت في
نفس الوقت لازم كلهم
يستأذنوا ال "master"



(I^2C) Inter Integrated Circuit (Cont.)

How can a Master send data to a specific Slave ?

Step	Direction	Message
1	Master -> Slave	Start
2	Master -> Slave	Slave Address
3	Master <- Slave	Ack
4	Master -> Slave	Data
5	Master <- Slave	Ack
6	Master -> Slave	Stop

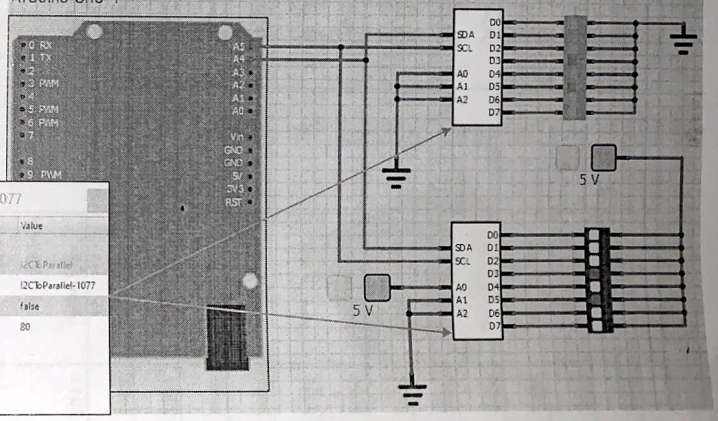


(I²C) Example(Port Expander)

```
#include <Wire.h>
#define IN_ADDRESS 80
#define OUT_ADDRESS 81
void I2C_scan(void){
    for (byte address=0;address<127;address++){
        Wire.beginTransmission(address);
        byte error=Wire.endTransmission();
        if (!error){
              Serial.print("I2c device is detected at address :");
              Serial.println(address);
        }
    }
}
void setup(){
    Wire.begin();
    Serial.begin(9600);
    I2C_scan();
}
void loop(){
      Wire.requestFrom(IN_ADDRESS, 1);
      byte read_val = Wire.read();
      Wire.beginTransmission(OUT_ADDRESS);
      Wire.write(read_val);
      Wire.endTransmission();
      delay(100);
}
```

```
I2c device is detected at address :80
I2c device is detected at address :81
```

Arduino Uno-4

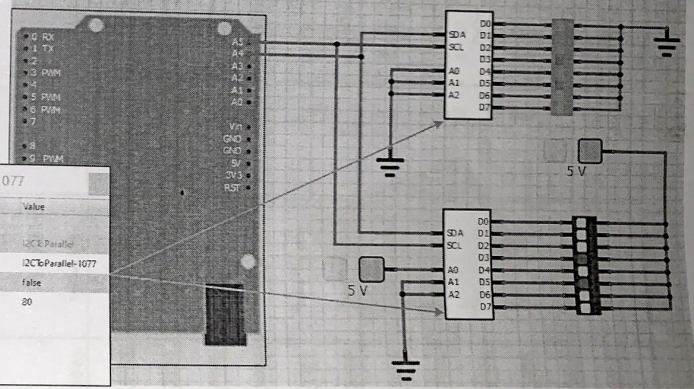


(I²C) Example(Port Expander)

```
#include <Wire.h>
#define IN_ADDRESS 80
#define OUT_ADDRESS 81
void I2C_scan(void){
    for (byte address=0;address<127;address++){
        Wire.beginTransmission(address);
        byte error=Wire.endTransmission();
        if (!error){
              Serial.print("I2c device is detected at address :");
              Serial.println(address);
        }
    }
}
void setup(){
    Wire.begin();
    Serial.begin(9600);
    I2C_scan();
}
void loop(){
        Wire.requestFrom(IN_ADDRESS, 1);
        byte read_val = Wire.read();
        Wire.beginTransmission(OUT_ADDRESS);
        Wire.write(read_val);
        Wire.endTransmission();
        delay(100);
}
```

```
I2c device is detected at address :80
I2c device is detected at address :81
```

Arduino Uno-4



I²C Externa EEPROM (Example) Cont.

في المثال ده بيكتب ٢٥٤ شوية ارقام في memory وبعدين يرجع يقرأها ويعرضها

```
#include <Wire.h>
const byte EEPROMAddress = 0x50;
char text[] = "2345432323454523452345234543254325432\n";
void I2CEEPROM_Write(unsigned int address, byte data) {
     Wire.beginTransmission(EEPROMAddress);
     Wire.send((int)highByte(address) );
     Wire.send((int)lowByte(address) );
     Wire.send(data);
     Wire.endTransmission();
     delay(5);
     // wait for the I2C EEPROM to complete the write cycle
}
byte I2CEEPROM_Read(unsigned int address) {
     byte data;
     Wire.beginTransmission(EEPROMAddress);
     Wire.send((int)highByte(address) );
     Wire.send((int)lowByte(address) );
     Wire.endTransmission();
     Wire.requestFrom(EEPROMAddress,(byte)1);
     while(Wire.available() == 0); // wait for data
     data = Wire.receive();
     return data;
}
void setup() {
     Wire.begin();
     for(int i=2;i<=5;i++) pinMode(i, OUTPUT);
     for(int i=0; i < sizeof(text); i++)
          I2CEEPROM_Write(i, text[i]);
}
void loop() {
     for(int i=0; i < sizeof(text); i++){
          char c = I2CEEPROM_Read(i);
          digitalWrite(c-'0', HIGH);
          delay(100);
          digitalWrite(c-'0', LOW);
     }
}
```

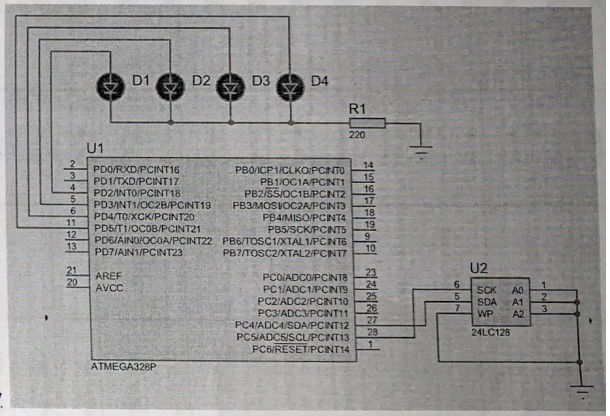
Wire.write(data)

Wire.read()

request From لو عايز اقرأ

لما بجب اكتب في الـ memory بعت ٣ حاجات الـ address مرتين والـ data مرة واحدة

لو عايز اقرأ data بعت الاول الـ address لي انا عايز اقرأ منك وبعدين اقوله request From



I²C LCD

```
#include <LiquidCrystal_AIP31068_I2C.h>
// set the LCD address to 0x3E for a 20 chars and 4 line display
LiquidCrystal_AIP31068_I2C lcd1(0x3E,20,2);
// set the LCD address to 0x3E for a 20 chars and 4 line display
LiquidCrystal_AIP31068_I2C lcd2(0x3F,20,2);

void setup()
{
  lcd1.init();                      // initialize the lcd
  lcd1.setCursor(3,0);
  lcd1.print("LCD1, Hello");
  lcd2.init();                      // initialize the lcd 2
  lcd2.setCursor(0,0);
  lcd2.print("HH:MM:SS");
  lcd2.setCursor(0,1);
  lcd2.print("  :  :  ");
}
int s=0;
int m=0;
int h=0;
void loop()
{
     h= (h<59)?  ((m==59)?h+1:h)  :0;
     m= (m<59)?  ((s==59)?m+1:m)  :0;
     s= (s<59)?s+1:0;
     lcd2.setCursor(6,1);
     lcd2.print(s);
     lcd2.setCursor(3,1);
     lcd2.print(m);
     lcd2.setCursor(0,1);
     lcd2.print(h);
     delay(1000);
}
```

3 : Colonne

هذا Cursor على السطر الأول

$(62)_{10} = (3E)_{16}$

Aip31068_i2c-722

Name	Value
Aip31068 i2c-722	
Xmltype	Aip31068_i2c
id	LCD1
Show id	true
Cols	16
Rows	2
Control Code	62

Arduino Uno-4

LCD1

LCD1, Hello

3F adresse

LCD2

HH:MM:SS
0 :0 :3

3F adresse

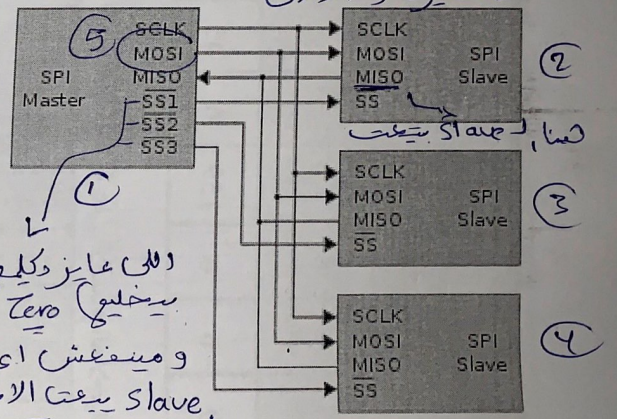
Data ⇒

Clock ⇒

83.22 kHz
19.04 kHz
0.000 Hz
0.000 Hz
Expand

(SPI) Serial Peripheral Interface

- Serial Peripheral Interface (SPI) is a synchronous serial data protocol used by microcontrollers for communicating with one or more peripheral devices quickly over short distances. It can also be used for communication between two microcontrollers.
- With an SPI connection there is always one master device (usually a microcontroller) which controls the peripheral devices. Typically, there are three lines common to all the devices:
  - MISO (Master In Slave Out) - The Slave line for sending data to the master,
  - MOSI (Master Out Slave In) - The Master line for sending data to the peripherals,
  - SCK (Serial Clock) - The clock pulses which synchronize data transmission generated by the master and one line specific for every device:
  - SS (Slave Select) - the pin on each device that the master can use to enable and disable specific devices.
- When a device's Slave Select pin is low, it communicates with the master. When it's high, it ignores the master. This allows you to have multiple SPI devices sharing the same MISO, MOSI, and CLK lines.
- *Arduino UNO (SCLCK : 13, MISO : 12, MOSI0 : 11, SS : 10)*



(SPI) Example (LCD SW/HW)

note: موجودة في البوسيلد

```
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Adafruit_PCD8544.h>

// Software SPI (slower updates, more flexible pin options):
// pin 7 - Serial clock out (SCLK)
// pin 6 - Serial data out (DIN)
// pin 5 - Data/Command select (D/C)
// pin 4 - LCD chip select (CS)
// pin 3 - LCD reset (RST)
Adafruit_PCD8544 sw_display = Adafruit_PCD8544(7, 6, 5, 4, 3);

// Hardware SPI (faster, but must use certain hardware pins):
// SCK is LCD serial clock (SCLK) - this is pin 13 on Arduino Uno
// MOSI is LCD DIN - this is pin 11 on an Arduino Uno
// pin 10 - Data/Command select (D/C)
// pin 9 - LCD chip select (CS)
// pin 8 - LCD reset (RST)
Adafruit_PCD8544 hw_display = Adafruit_PCD8544(10, 9, 8);
// Note with hardware SPI MISO and SS pins aren't used but will still be read
// and written to during SPI transfer.  Be careful sharing these pins!

void setup()   {
  sw_display.begin();
  sw_display.clearDisplay();   // clears the screen and buffer
  sw_display.drawPixel(10, 10, BLACK);
  sw_display.display();

  hw_display.begin();
  hw_display.clearDisplay();   // clears the screen and buffer
  hw_display.drawPixel(30, 30, BLACK);
  hw_display.display();
}

void loop() {
}
```

Software display

hardware display