Intro to IOT

2017, Demonstration 2

1. Background: IoT architecture

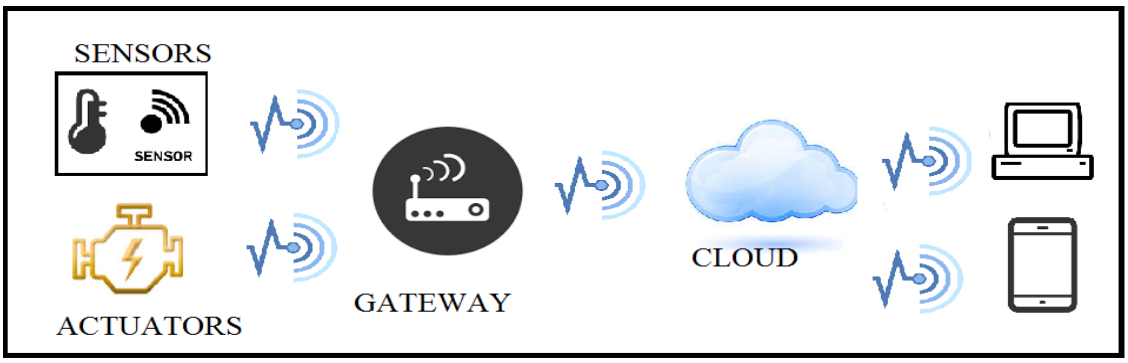
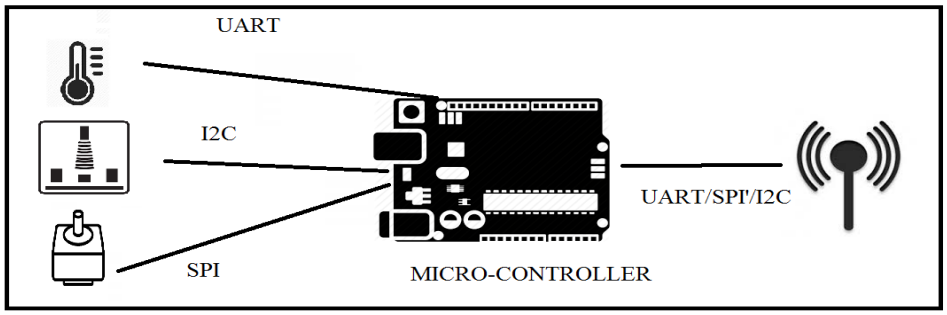


Fig. 1: IoT architecture

An IoT system includes:

- Sensor nodes: Sensors and actuators
- Gateways
- Cloud + back end

a) Sensor nodes



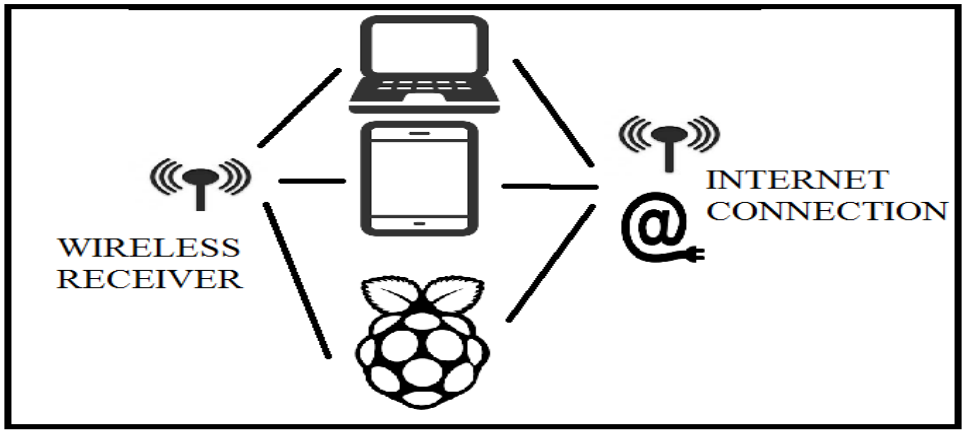
SENSOR DEVICE STRUCTURE

Fig. 2: Sensor device structure

A sensor node includes

- Sensors/Actuators
- Microcontroller
- Wireless transceiver module

b) Gateways



GATEWAY STRUCTURE

A gateway includes:

- Wireless receiver
- Micro-controller
- Internet connection

c) Cloud + back-end

Cloud servers + the back-end system are for storing data, hosting web pages and presenting data

2. Demonstration

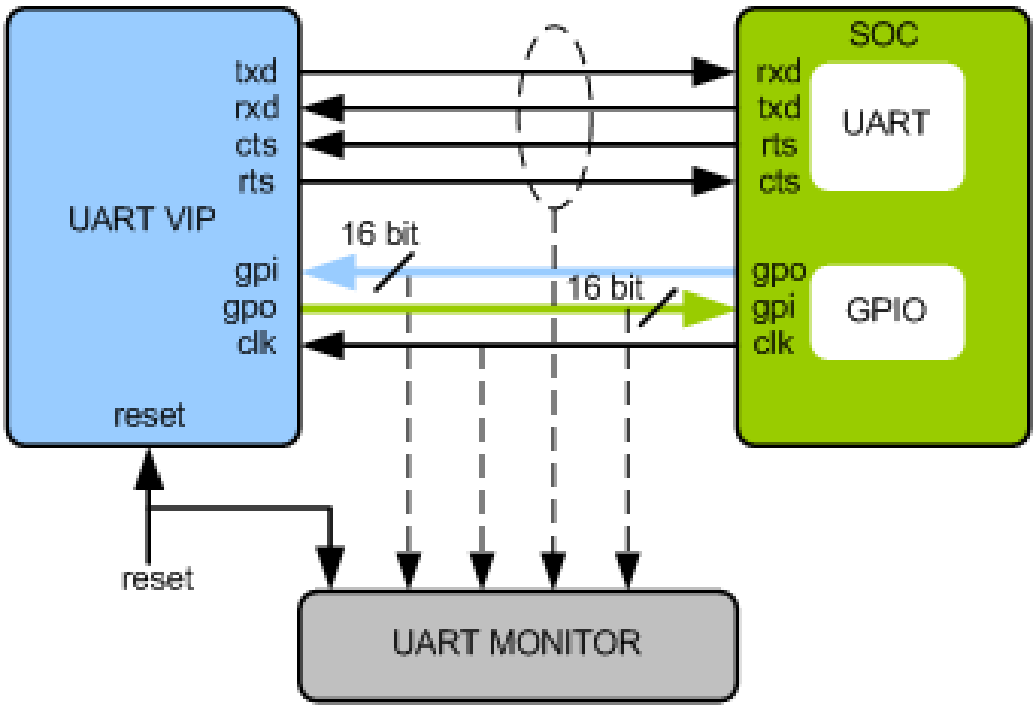
In this section, we learn:

- to set up connections between sensors and a micro-controller
- to set up a connection between a micro-controller and a wireless transceiver module
- to set up a wireless connection between a sensor node and a gateway

There are three primary wire communication protocols:

- UART
- SPI
- I2C

a) UART



- An UART port mainly includes one input port and one output port.
- The number of UART ports depends on particular micro-controllers.
- For 2 ways communication : 1 UART port is often used for connecting to a single devices
- For one-way communication (output): an output port can be connected to many devices to control all of them simultaneously. For example, one output port can be used to control many LEDs.

- In addition to hardware UART, software UART ports can be defined and used. Fortunately, you have software UART integrated in Arduino IDE.

Example of hardware UART:

```
byte byteRead;
void setup() {
// Turn the Serial Protocol ON
  Serial.begin(9600);
}

void loop() {
   /*  check if data has been sent from the computer: */
  if (Serial.available()) {
     /* read the most recent byte */
     byteRead = Serial.read();
     /*ECHO the value that was read, back to the serial port. */
     Serial.write(byteRead);
  }
}
```

Example of software UART

```
#include <SoftwareSerial.h>// import the serial library
SoftwareSerial Genotronex(10, 11); // RX, TX
int ledpin=13; // led on D13 will show blink on / off
int BluetoothData; // the data given from Computer

void setup() {
  // put your setup code here, to run once:
  Genotronex.begin(9600);
  Genotronex.println("Bluetooth On please press 1 or 0 blink LED ..");
  pinMode(ledpin,OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
   if (Genotronex.available()){
BluetoothData=Genotronex.read();
   if(BluetoothData=='1'){   // if number 1 pressed ....
   digitalWrite(ledpin,1);
   Genotronex.println("LED  On D13 ON ! ");
   }
  if (BluetoothData=='0'){// if number 0 pressed ....
  digitalWrite(ledpin,0);
   Genotronex.println("LED  On D13 Off ! ");
  }
}
delay(100);// prepare for next data ...
}
```

b) SPI

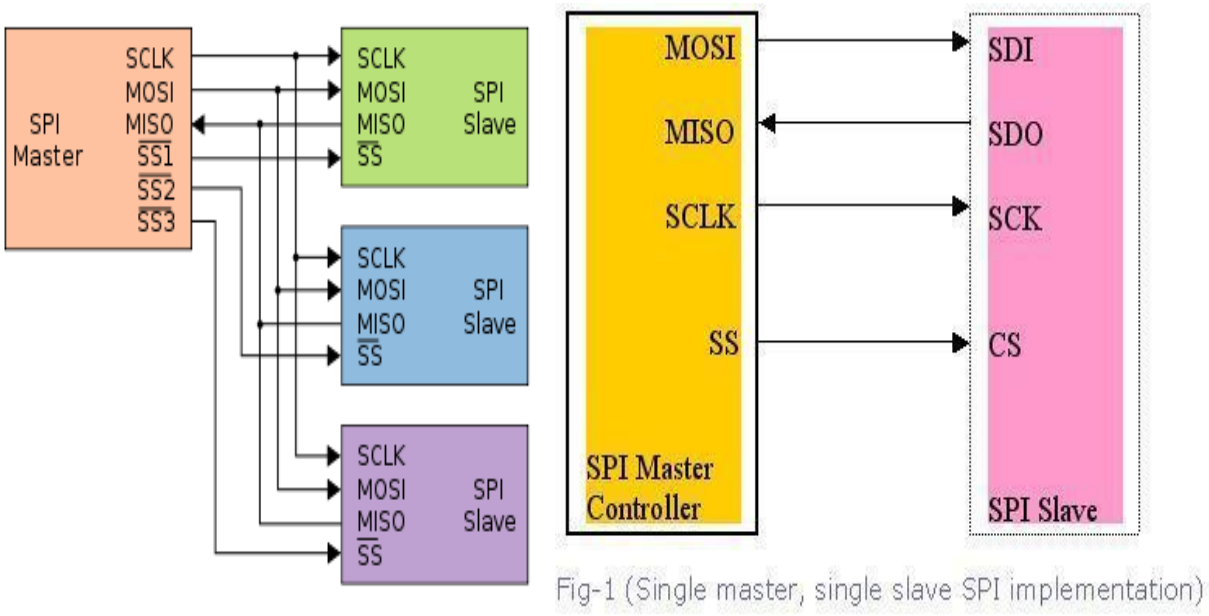


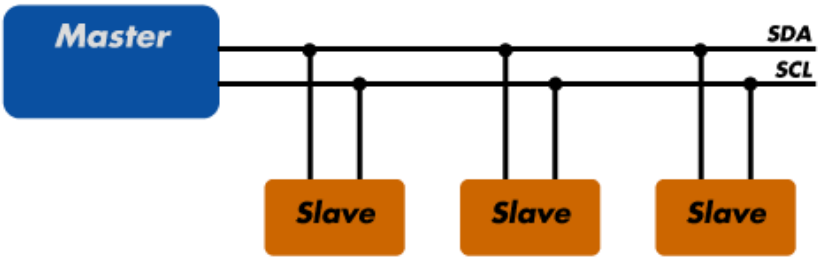
Fig-1 (Single master, single slave SPI implementation)

- An SPI port mainly includes four ports: SLK, MOSI, MISO, and SS. The SPI master can have more than one SS port. For example, SPI master has 3 SS ports in the picture above
- The number of SPI ports depends on particular micro-controllers.
- For 2 ways communication : 1 UART port can be connected to a single or multiple devices
- The SPI master choose the target slave by activate (active low) the SS pin of the device
- In Arduino use library #include <SPI.h>

```
#include "SPI.h" // necessary library
int ss=10; // using digital pin 10 for SPI slave select
int del=200; // used for various delays

void setup()
{
  pinMode(ss, OUTPUT); // we use this for SS pin
  SPI.begin(); // wake up the SPI bus.

  }

void setValue(int value)
{
  digitalWrite(ss, LOW);
  SPI.transfer(0); // send command byte
  SPI.transfer(value); // send value (0~255)
  digitalWrite(ss, HIGH);
}

void loop()
{
  for (int a=0; a<256; a++)
```

```
  {
    setValue(a);
    delay(del);
  }
  for (int a=255; a>=0; --a)
  {
    setValue(a);
    delay(del);
  }
}
```

c) I2C



- I2C port mainly includes SDA and SCL
- SDA (data line) and SCL (clock line)
- Master chooses its slave via an address of a slave.
- In adruino use library: <Wire.h>
- Master I2C reads data from a slave

```
#include <Wire.h>
int nodePayload[PAYLOAD_SIZE];
void setup()
{
Wire.begin();
}
void loop()
{
Wire.requestFrom(nodeAddress, PAYLOAD_SIZE);    // request data from node#
    if(Wire.available() == PAYLOAD_SIZE) {  // if data size is avaliable from nodes
      for (int i = 0; i < PAYLOAD_SIZE; i++)
                    nodePayload[i] = Wire.read();  // get nodes data
}
```

3. Hands-on:

The demonstration includes three exercises. For all the three parts, you need to stack the grove shield on top of the Arduino because LCD needs to be connected to grove shield.

A. Get familiar with I2C and control LCD

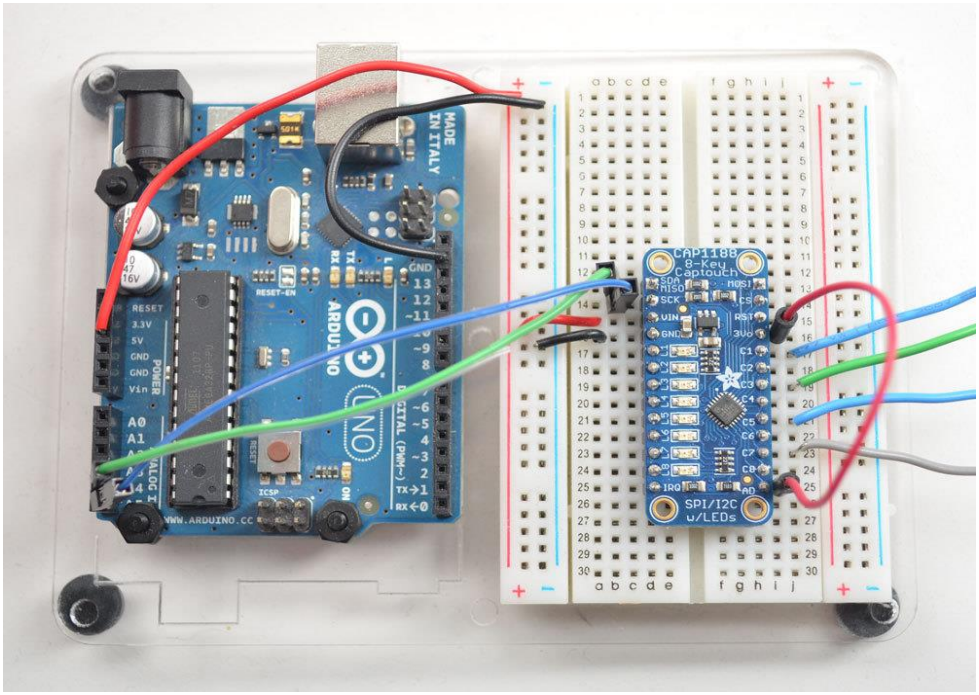
Connect LCD to Arduino and write “hello world” on it using I2C. You can find the information about LCD as well as library and sample code here: http://wiki.seeed.cc/Grove-LCD_RGB_Backlight/

B. read from touch sensor

I2C is very useful when you need to connect multiple sensors and actuators to your controller with a limited number of pins. The sensors/actuators will share the pins as long as each one has a unique address. I2C is not very fast, but will be proper for many sensors/actuators.

Your task is to connect touch sensor to Arduino via I2C. Then write a program to show the number of touched sensors on LCD. E.g. if I touch two pins, it has to show 2, otherwise 0. Note that you have to touch the C pins on the touch sensor not the L pins.

You can always use the serial port for debugging. You can follow this wiring.



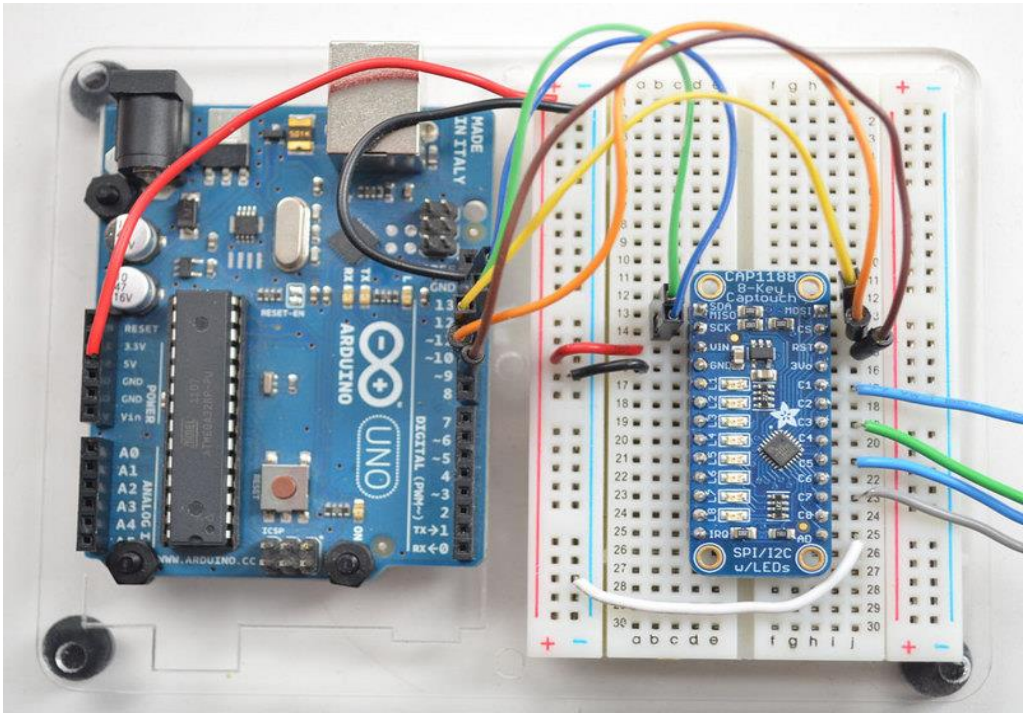
Sensors/actuators always come with a default address, and it might be possible to change it. Connecting AD to 3Vo will enable the I2C with the default address of 0x28. You can use the example code for LCD and touch sensor.

For more information, refer to https://learn.adafruit.com/adafruit-cap1188-breakout/overview

Question: Explain why multiple I2C devices can be connected to the same I2C port.

A. Get familiar with SPI

Repeat part B with SPI.



Optional: on LCD, show the pin number that has touched.