ECOLE NATIONALE
SUPÉRIEURE
D'INFORMATIQUE

الجـــمهوريـــة الجزائريـــة الديمـــقراطيـــة الشـعبيـة
وزارة التعليـــم العـــالي والبـــحث العلمـــي
People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

Intelligent and Communicating Systems, ICS

2^{nd} Year Specialty SIQ G02, 2CS SIQ2

LAB report n°07

Title:

Arduino-Raspberry Wired Communications I2C

Studied by:

First Name: Nour el Imane
Last Name: HEDDADJI
E-mail: jn_heddadji@esi.dz

A. Theory

1. Theoretical study on I2C and SPI communication and comparis SPI, I2C, and UART protocols

I2C, SPI, and UART are three distinct serial communication protocols widely employed in interconnecting electronic devices.

I2C, or Inter-Integrated Circuit, operates synchronously and utilizes a two-wire interface : a serial data line (SDA) and a serial clock line (SCL). It is well-suited for low-speed data transfer applications, such as sensor interfaces and low-power devices. I2C supports a device addressing scheme, making it ideal for communication with multiple devices on the same bus.

SPI, or Serial Peripheral Interface, is another synchronous communication protocol that employs a four-wire interface: **MOSI (Master Out Slave In), MISO (Master In Slave Out), SCLK (Serial Clock),** and SS (Slave Select). SPI is commonly used in applications demanding high-speed data transfer, like memory cards, LCD displays, and sensors. It facilitates communication with multiple devices simultaneously through distinct chip select lines.

UART, or Universal Asynchronous Receiver/Transmitter, is on the other hand an asynchronous communication protocol employing a two-wire interface with a transmit line (TX) and a receive line (RX). UART is prevalent in applications requiring long-range communication, such as GPS, wireless communication, and satellite communication. Unlike I2C and SPI, UART can communicate with only one device at a time and doesn't require a shared clock signal, simplifying its implementation.

lightgray **Characteristic**	**I2C**	**SPI**	**UART**	**Remarks**
Speed	Slower	Faster	Slower	SPI is the fastest.
Number of Devices	Multiple at the same time	Multiple at the same time	Single	I2C and SPI support multiple devices. UART is point-to-point.
Wiring Complexity	Fewer wires	More wires	More wires	I2C requires fewer wires.
Clocking	Synchronous	Synchronous	Asynchronous	I2C and SPI use synchronous clocking, while UART uses asynchronous clocking.
Error Checking	Yes	Limited	Limited	I2C has built-in error checking mechanisms.

Table 1: Comparison of I2C, SPI, and UART

2. Theoretical study of I2C of Raspberry pins and software related to I2C and to Analog-to-Digital Converter ADS1115.

2.1. I2C Communication on Arduino MKR 1010

- **I2C Pins:** On the Arduino MKR 1010, the I2C pins are labeled SDA (Serial Data) and SCL (Serial Clock).
- **Pin Locations**: SDA is located on pin 11, and SCL is located on pin 12.
- **Built-in I2C Hardware:** These pins are connected to the microcontroller's built-in I2C hardware module, facilitating I2C communication.
- **Wire Library:** The Arduino IDE includes a built-in Wire library for I2C communication. This library supports both master and slave communication, allowing the microcontroller to function as an I2C master or slave.

2.2. ADS1X15 Library for ADC:

The ***ADS1X15 Library*** is used for configuring the ADC, specifically the ADS1115.

- **Configuration Options:** It provides functions to configure the ADC, including setting the input channel, gain, and data rate. These settings can be adjusted to meet the requirements of your application.

- **Reading Analog Values:** To read analog values from the ADS1115, the library includes functions to initiate a conversion and read the digital value from the ADC.
- **Unit Conversion:** Additionally, the library offers functions to convert the digital value to a voltage or other physical units. This conversion is based on the input range and other settings of the ADC.

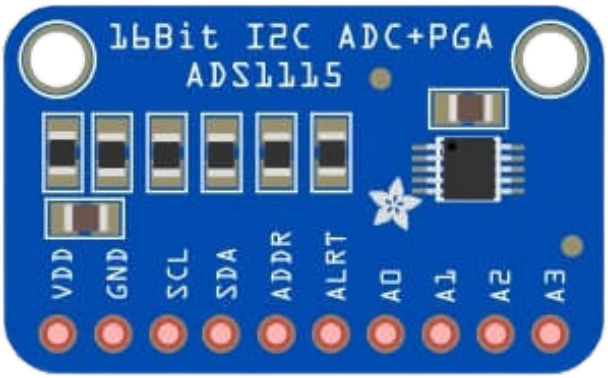


Figure 1: ADS1X15

3. Theoretical study of I2C of Raspberry pins and software (Library) relate ADS1115

3.1. I2C Communication on Raspberry Pi

- **I2C Pins:** The Raspberry Pi has dedicated I2C pins - SDA (Serial Data) and SCL (Serial Clock).
- **Pin Labels:** SDA is labeled as GPIO2 (BCM 2), and SCL is labeled as GPIO3 (BCM 3) on the Raspberry Pi.
- **Data Transmission:** These pins are used for transmitting and receiving data over the I2C bus.
- To use the I2C bus on the Raspberry Pi, you need to enable it using the raspi-config tool or by editing the config.txt file.
- Once enabled, the I2C bus can be used to communicate with various devices and sensors.

3.2. I2C Libraries for Raspberry Pi

- Several libraries are available for the Raspberry Pi to interface with I2C devices, such as smbus, wiringPi, pigpio, etc.

ADS1115 Library for Raspberry Pi

- The ADS1115 16-bit analog-to-digital converter (ADC) is commonly used with the Raspberry Pi.
- Libraries such as Adafruit Python ADS1X1 are available for configuring and reading the ADC using the I2C bus.
- Functions in the library include setting the input channel, gain, data rate, and reading raw/converted values from the ADC.

3.3. Raspberry Pi - I2C (with CAN ADS1115)

To configure I2C on the Raspberry Pi, follow these steps:

1. Enable I2C in Raspberry Pi Configuration:
    1. Open the terminal and type `sudo raspi-config`.
    2. This will open the Raspberry Pi Configuration tool.
    3. Navigate to "Interface Options" and select "I2C".
    4. Choose "Yes" to enable the I2C interface.
    5. Select "Finish" to exit.

1. Install the necessary packages:

```
sudo apt-get install -y python-smbus i2c-tools
```

2. Add the user to the I2C group:

```
sudo adduser pi i2c
```

3. Reboot your Raspberry Pi:

```
sudo reboot
```

4. Check I2C device:

```
sudo i2cdetect -y 1
```

This command will display a grid of numbers representing the addresses of connected I2C devices. If there are no connected devices, all cells will display "–". If there are connected devices, their addresses will be displayed.

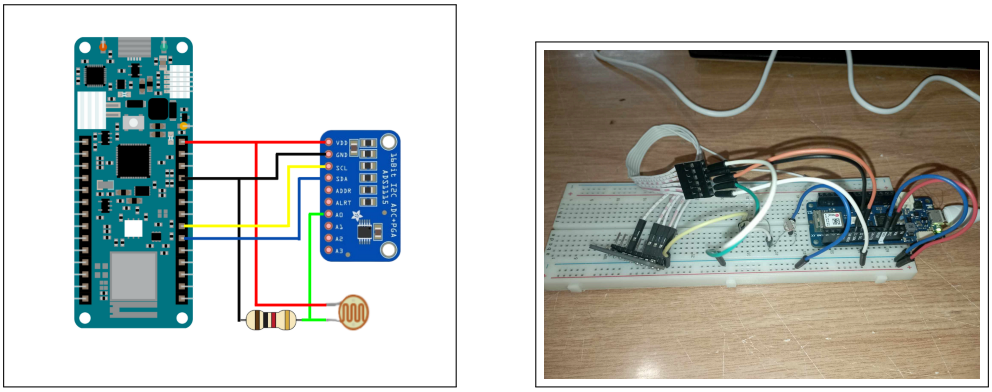
B. Activity

1. Arduino-I2C (with CAN ADS1115)

1.1. Using LDR sensor

Hardware

The setup involves a basic I2C connection between an Arduino and ADS1115. The ADS is connected to an LDR sensor.



(a) Schematic view (b) Circuit view

Figure 2: Arduino-I2C and LDR connection electronics views

Software

The light value is read by the ADS1115 using the function read.ADC() and then sent to the arduino to be printed

```
#include <ADS1X15.h>
ADS1115 ADS(0x48);
void setup() {
  Serial.begin(9600);
  ADS.begin();
  ADS.setGain(0);
  ADS.setMode(1);
  ADS.setDataRate(7);
  ADS.readADC(0);
}
void loop() {
  int16_t tension_A0 = ADS.readADC(0);
  // Mapping the value
  byte valMap = map(tension_A0, 0, 32767, 0, 255);
  Serial.print("Light value: ");
  Serial.println(valMap);
```

```
17    delay(100);
18  }
```

Listing 1: Arduino-I2C and LDR connection program

Result

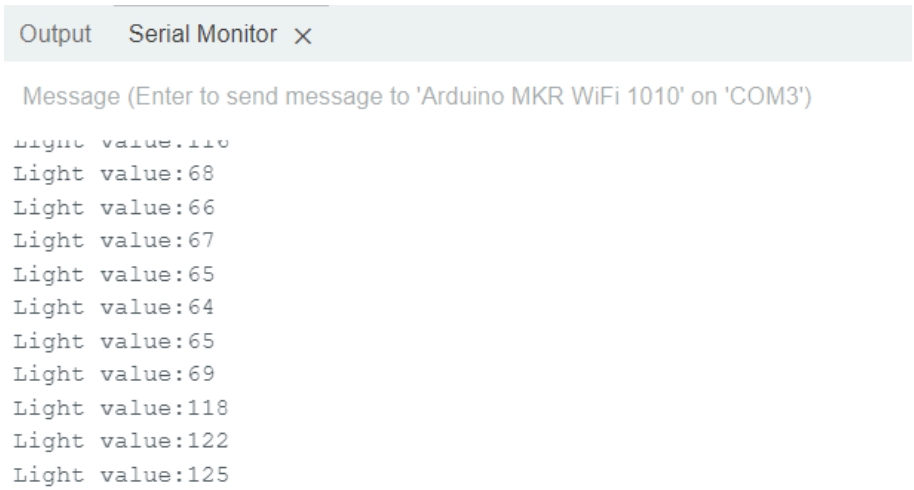
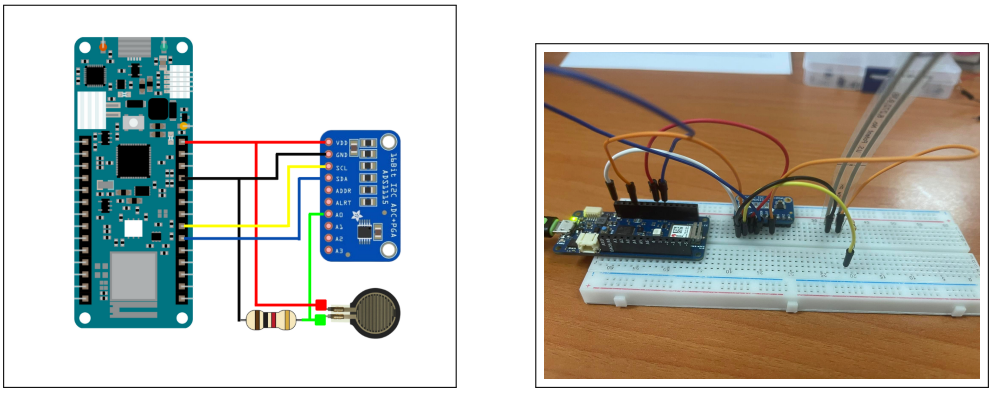


Figure 3: console view

1.2. Using Force Sensor

Hardware

The setup involves a basic I2C connection between an Arduino and ADS1115. The ADS is connected to an Force sensor.



(a) Schematic view (b) Circuit view

Figure 4: Arduino-I2C and Force Sensor connection electronics views

Software

The light value is read by the ADS1115 using the function read.ADC() and then sent to the arduino to be printed

```
#include <ADS1X15.h>
ADS1115 ADS(0x48);
void setup() {
  Serial.begin(9600);
  ADS.begin();
  ADS.setGain(0);
  ADS.setMode(1);
  ADS.setDataRate(7);
  ADS.readADC(0);
}
void loop() {
  int16_t tension_A0 = ADS.readADC(0);
  Serial.print("Force value: ");
  Serial.println(valMap);
  delay(100);
}
```

Listing 2: Arduino-I2C and Forse Sensor connection program

Result

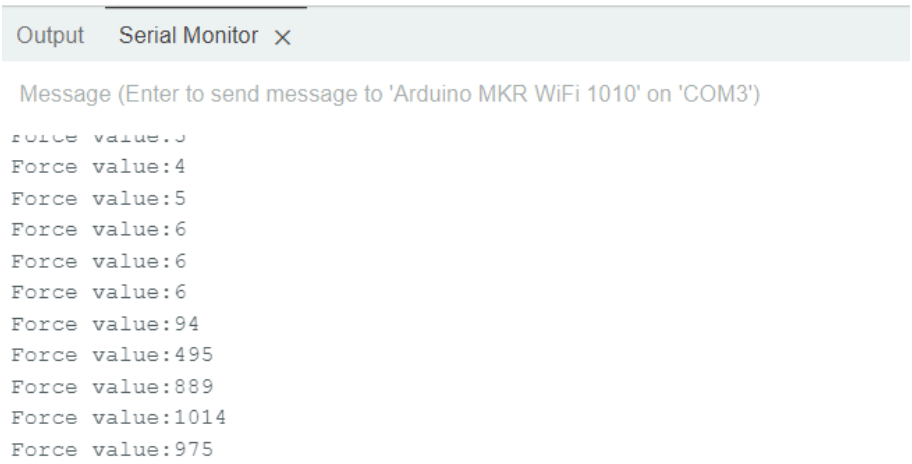


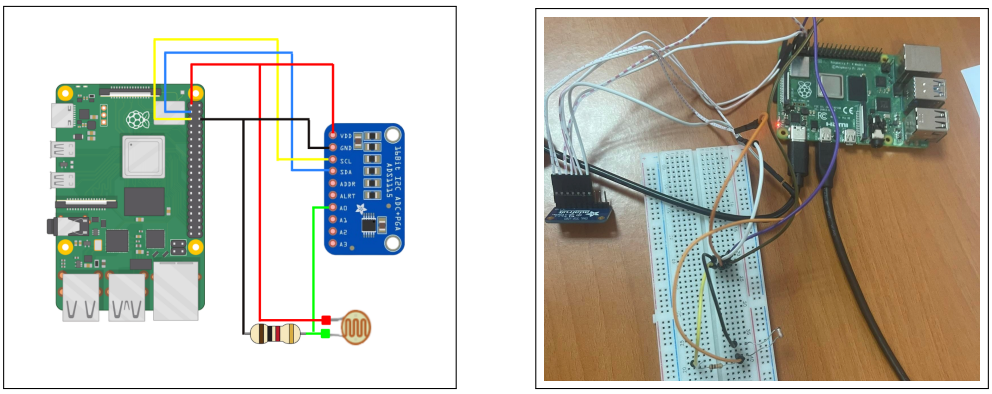
Figure 5: Console View

2. Raspberry-I2C (with CAN ADS1115)

2.1. Using LDR sensor

Hardware

The setup involves a basic I2C connection between a Raspberry and ADS1115. The ADS is connected to an LDR sensor.



(a) Schematic view (b) Circuit view

Figure 6: Raspberry-I2C and LDR connection electronics views

Software

The light value is read by the ADS1115 using the function read.ADC() and then sent to the Raspberry to be printed

```
import Adafruit_ADS1x15
from time import sleep
adc = Adafruit_ADS1x15.ADS1115(address=0x48, busnum=1)
GAIN = 1  # Set gain to 1
adc.start_adc(0, gain=GAIN)
while True:
    value = adc.read_adc(0, gain=GAIN)  # Read the force value
    print("Light Value:", value)
    sleep(0.1)
```

Listing 3: Raspberry-I2C and LDR connection program

Result

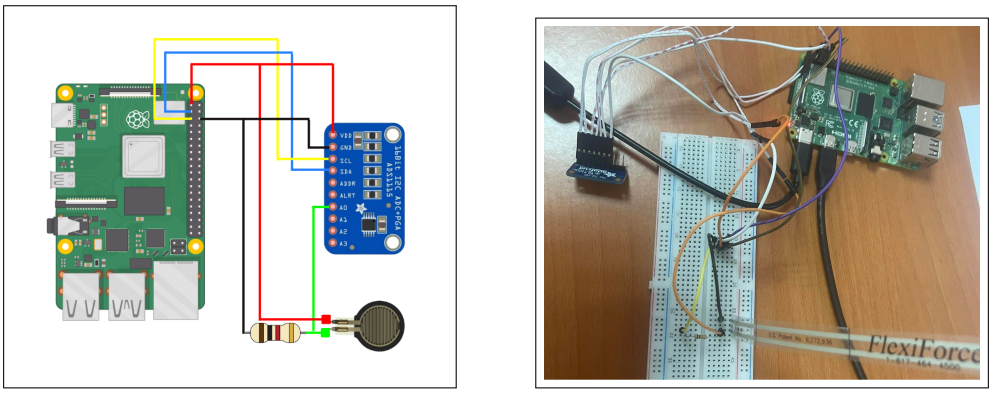
```
Shell
Light Value :  14062
Light Value :  13968
Light Value :  13920
Light Value :  13933
Light Value :  13801
Light Value :  13608
Light Value :  13793
Light Value :  13981
Light Value :  14076
Light Value :  13945
Light Value :  13274
Light Value :  10876
Light Value :  7135
```

Figure 7: console view : Result

2.2. Using Force Sensor

Hardware

The setup involves a basic I2C connection between a Raspberry and ADS1115. The ADS is connected to an Force sensor.



(a) Schematic view (b) Circuit view

Figure 8: Raspberry-I2C and Force Sensor connection electronics views

Software

The light value is read by the ADS1115 using the function read.ADC() and then sent to the arduino to be printed

```python
import Adafruit_ADS1x15
from time import sleep
adc = Adafruit_ADS1x15.ADS1115(address=0x48, busnum=1)
GAIN = 1  # Set gain to 1
adc.start_adc(0, gain=GAIN)
while True:
    value = adc.read_adc(0, gain=GAIN)  # Read the force value
    print("Force Value:", value)
    sleep(0.1)
```

Listing 4: Raspberry-I2C and LDR connection program

Result

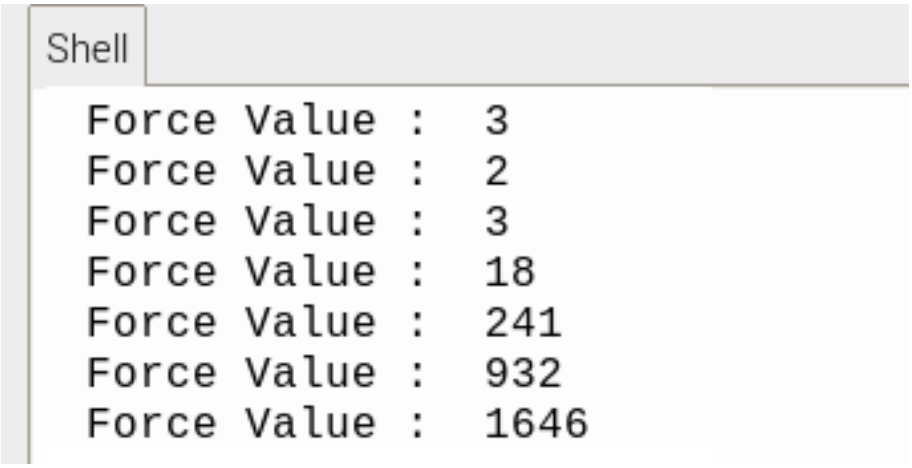


Figure 9: Console View

B. Conclusion

In conclusion, the adoption of I2C communication in both Arduino and Raspberry Pi environments proves to be an effective strategy for connecting multiple devices. The I2C protocol's simplicity, requiring just two wires for communication, and its ability to support numerous devices on the same bus make it a versatile choice. The external Analog-to-Digital Converter (ADC), exemplified by the ADS1115, enhances the capabilities of both platforms, providing finer control over the analog-to-digital conversion process.

Integrating an external ADC with Arduino enables advantages like increased resolution and enhanced accuracy in analog signal measurements, crucial for precision-centric applications. For Raspberry Pi, leveraging an external ADC like the ADS1115 is indispensable for reading analog inputs from sensors.