Experiment No. 02:

Motion Sensing System using Accelerometer and Raspberry Pi

By: Abhilash Kashyap Balasubramanyam

Lab Project Partner: Adnan Patel

Instructor: Dr. Jafar Saniie

ECE 510

Lab Date: 05-23-2024

Due Date: 05-31-2024

Acknowledgment: I acknowledge all of the work (including figures and codes) belongs to me and/or persons who are referenced.

Signature: 

I. Introduction

A. Purpose

Building a motion sensing system with an accelerometer and a Raspberry Pi is required for this lab. Client-server communication is used to send the sensor data to a server. Data that has been encrypted is transferred from the Raspberry Pi to the server via a UDP connection, where it is decrypted and visually displayed.

B. Background

Raspberry Pi

The Raspberry Pi Foundation is a British company that developed the Raspberry Pi line of small single-board computers. Promoting the teaching of fundamental computer science in classrooms and in underdeveloped nations was the goal. The initial model sold outside of its designated market for applications like robotics and became significantly more popular than expected. Cases and peripherals (such keyboards and mouse) are not included. However, several official and unofficial packages have included certain accessories. A System on a Chip (SoC), or a whole computer on a single chip, is what powers the Raspberry Pi. It includes a CPU, graphics processing unit (GPU), memory, and all other required parts. The Broadcom BCM2835 SoC, which powers the Raspberry Pi, has an ARM1176JZF-S 700 MHz CPU, a VideoCore IV GPU, and 512 MB of RAM.

The General-Purpose Input/Output (GPIO) pins of the Raspberry Pi are one of its most important features. These serve as the Pi's physical interface with the outside world. They can be thought of as switches, either input or output, at their most basic level. Beyond that, though, GPIO pins can communicate with a vast array of parts, from basic LEDs and buttons to intricate sensors, displays, and even the internet.

Forty GPIO pins of the Raspberry Pi are used to connect motors, lights, sensors, and other devices. Compared to the original Raspberry Pi's 26-pin GPIO header, this is a major gain. Every pin is unique and has multiple applications, including as GPIO, I2C, SPI, UART, and more. On top of the Raspberry Pi, close to the yellow video out socket, are the GPIO pins. There are two rows of twenty pins each, organized in a 2x20 layout. This facilitates the connection of a large variety of accessories and gadgets to the Raspberry Pi.

In addition, the Raspberry Pi features several connectors that let it establish connections with other gadgets and the internet. These consist of an SD card slot, an Ethernet port, an HDMI port, and a USB port. While the HDMI port can be used to connect a display, the USB port is used to attach accessories like a keyboard and mouse. You can load the operating system and save data on the SD card slot in addition to connecting to the internet via the Ethernet connector. To sum up, the Raspberry Pi is an incredibly strong and adaptable gadget that has a plethora of uses. It is an excellent tool for learning computer science and electronics because of its GPIO pins and ports, which enable it to communicate with a variety of devices and the internet.

Adafruit ADXL345 Triple-Axis Accelerometer

A small, low-power, triple-axis accelerometer with digital I2C and SPI interface breakout is the Adafruit ADXL345. At up to ±16g, it offers high-resolution (13-bit) readings. In tilt-sensing applications, this sensor may measure the static acceleration of gravity as well as the dynamic acceleration brought on by motion or shock. It can detect inclination variations of less than 1.0 degree because to its excellent resolution. The ADXL345 is perfect for applications that need to monitor dynamic acceleration brought on by motion or shock, as well as gravity in tilt-sensing

applications. It is appropriate for mobile applications due to its tiny size and low power consumption.

I^2C (Inter Integrated Circuit) Bus

Philips Semiconductor (now NXP Semiconductors) designed the I2C (Inter-Integrated Circuit) Bus, a packet-switched, single-ended, multi-master, multi-slave serial computer bus. It is frequently used in short-range intra-board communication to link slower peripheral integrated circuits to processors and microcontrollers. The Serial Data Line (SDA) and Serial Clock Line (SCL), which are pushed up with resistors, are the only two bidirectional open-drain lines used by the I2C bus. It is a well-liked option for many kinds of electronic gadgets due to its simplicity and versatility.

SPI (Serial Peripheral Interface) Bus

A synchronous serial communication interface specification used for short-distance communication, mostly in embedded devices, is called the SPI (Serial Peripheral Interface) Bus. It was created by Motorola and is now considered the de facto standard. Liquid crystal displays and Secure Digital cards are typical examples of uses. SPI devices use a master-slave design with a single master to communicate in full duplex mode. The reading and writing frames are created by the master device. Individual slave chooses lines allow for the selection of multiple slave devices.

Server-Client Architecture

In a server-client architecture, every machine or process connected to the network functions as either a server or a client. Dedicated to controlling disc drives (file servers), printers (print servers),

or network traffic (network servers), servers are strong computers or programs. Workstations or PCs that users run apps on are known as clients. For resources like data, devices, and even processing power, clients rely on servers.

Encryption of Electronic Data

The process of transforming data into a format that is difficult for unauthorized parties to decipher is known as encryption. It is employed to safeguard private data that is sent or kept on electronic devices. The method converts the readable data (plaintext) into an encoded piece of information (ciphertext) using an algorithm and a key. After that, the information must be decoded and put back into its original form using the key. Encryption is an essential technique for protecting sensitive data's confidentiality, stopping illegal access, and preserving the security and integrity of data.

II. Lab Procedure and Equipment List

A. Equipment

Equipment

- 1 x Raspberry Pi Single Board Computer
- 1 x Adafruit ADXL345 Triple-Axis Accelerometer
- Linux Development Environment
- I^2C (Inter-Integrated Circuit) Bus
- SPI (Serial Peripheral Interface)
- Client-Server Internet Communication Architecture

B. Procedure

Implementation of Real-Time Motion Sensing Device using I^2C

1. Install the Raspbian operating system on the Raspberry Pi.
2. Open Terminal, enter the interface commands of the Raspberry Pi and enable I^2C and SPI.

3. Connect the accelerometer along with the Raspberry Pi necessarily as per the lab manual.
4. Compile the codes main.cpp and ADXL345.cpp.
5. Run the program and record the results displayed both on the terminal and the output file created in the same depository.

Implementation of Real-Time Motion Sensing Device using SPI and Client Server Communication

1. Open Terminal, enter the interface commands of the Raspberry Pi and enable I^2C, SPI and SSH.
2. Connect the accelerometer along with the Raspberry Pi necessarily as per the lab manual.
3. Compile the code client.cpp on the Raspberry Pi and the server.py on the secondary device.
4. Run the codes on both the client (raspberry pi) and server (secondary device) and record the observations.
5. The changes to the original position of the accelerometer is both recorded and can be observed figuratively and mathematically.

III. Results and Demonstration

The demo video is available at this Link. Following are the screenshots from the experiment.

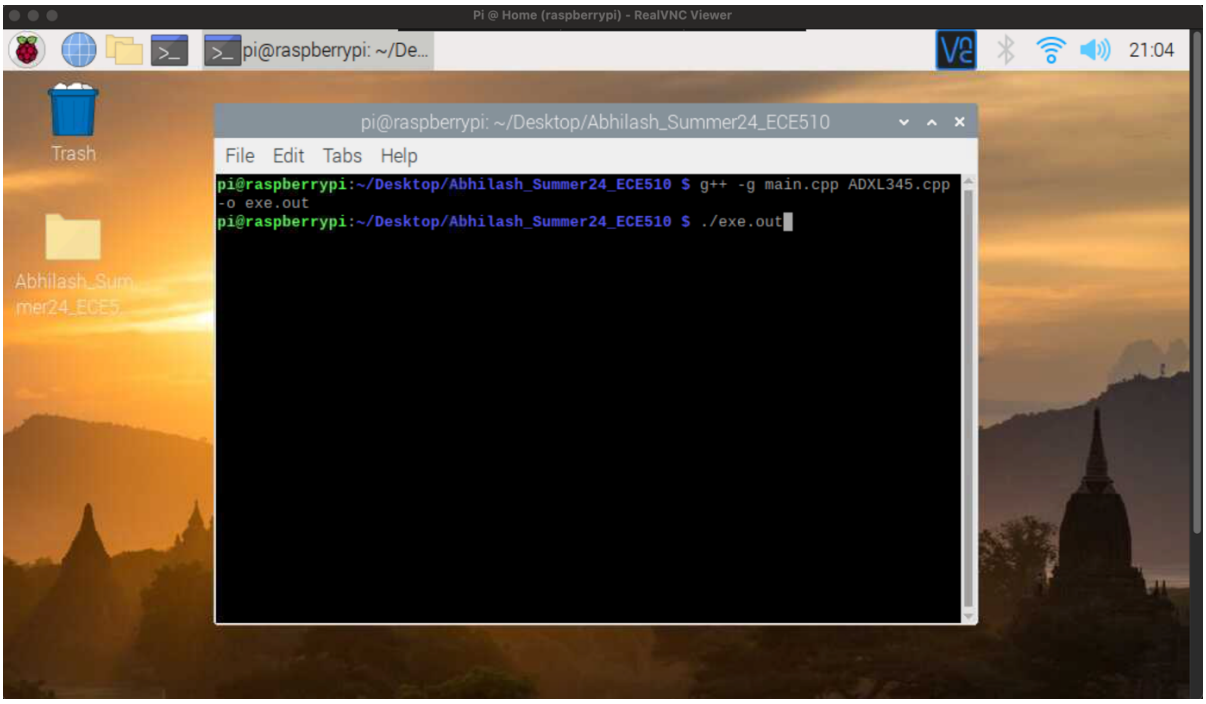

Figure 1: Compilation and Run of Code on Terminal.

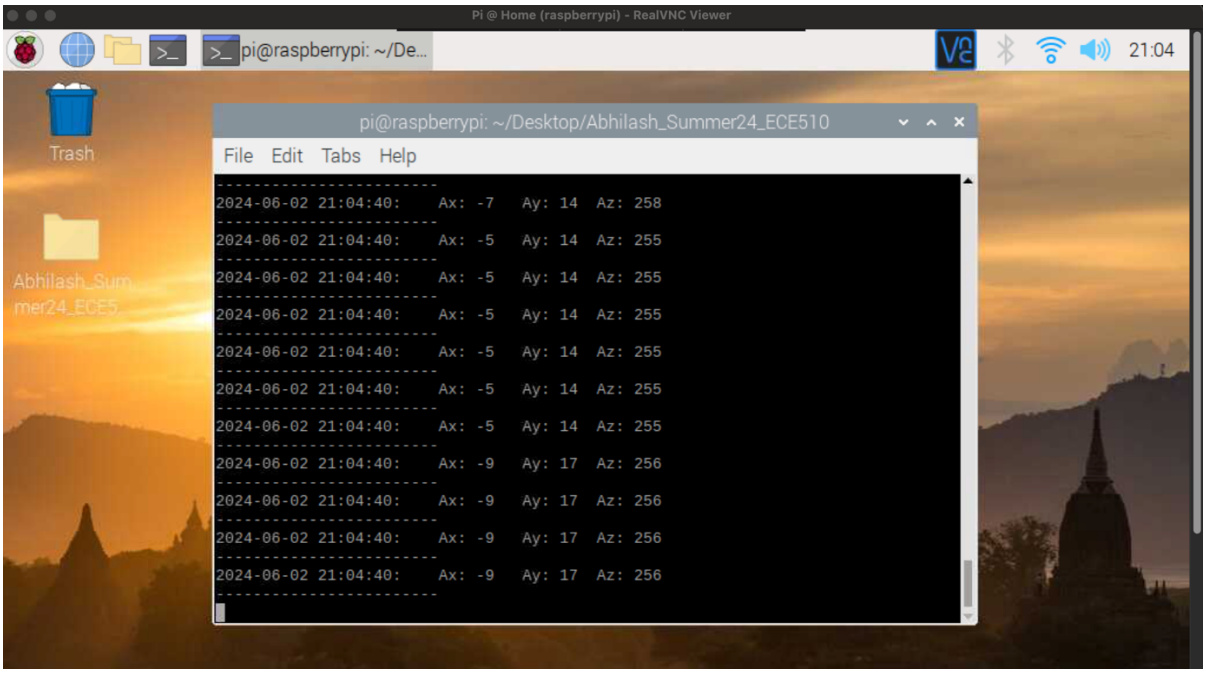

Figure 2: Display of Results on the Terminal.

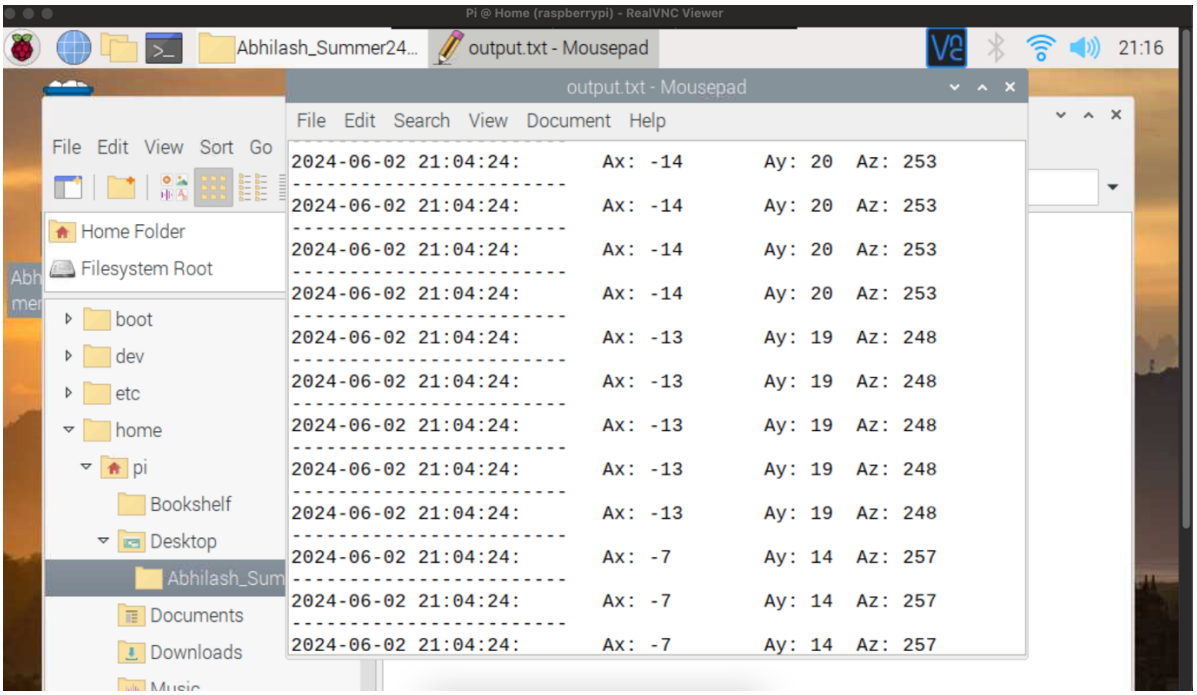

Figure 3: Display of Results in the stored Output file.

Implementation of Real-Time Motion Sensing Device using SPI and Client Server Communication

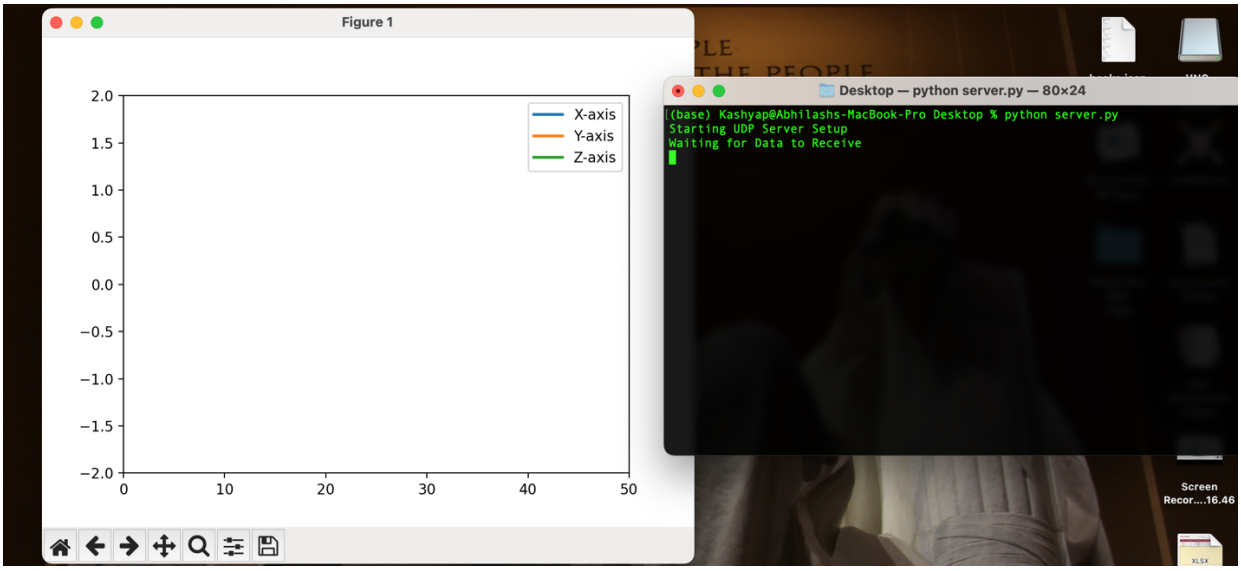

Figure 4: Server Startup and Listening to the Client.

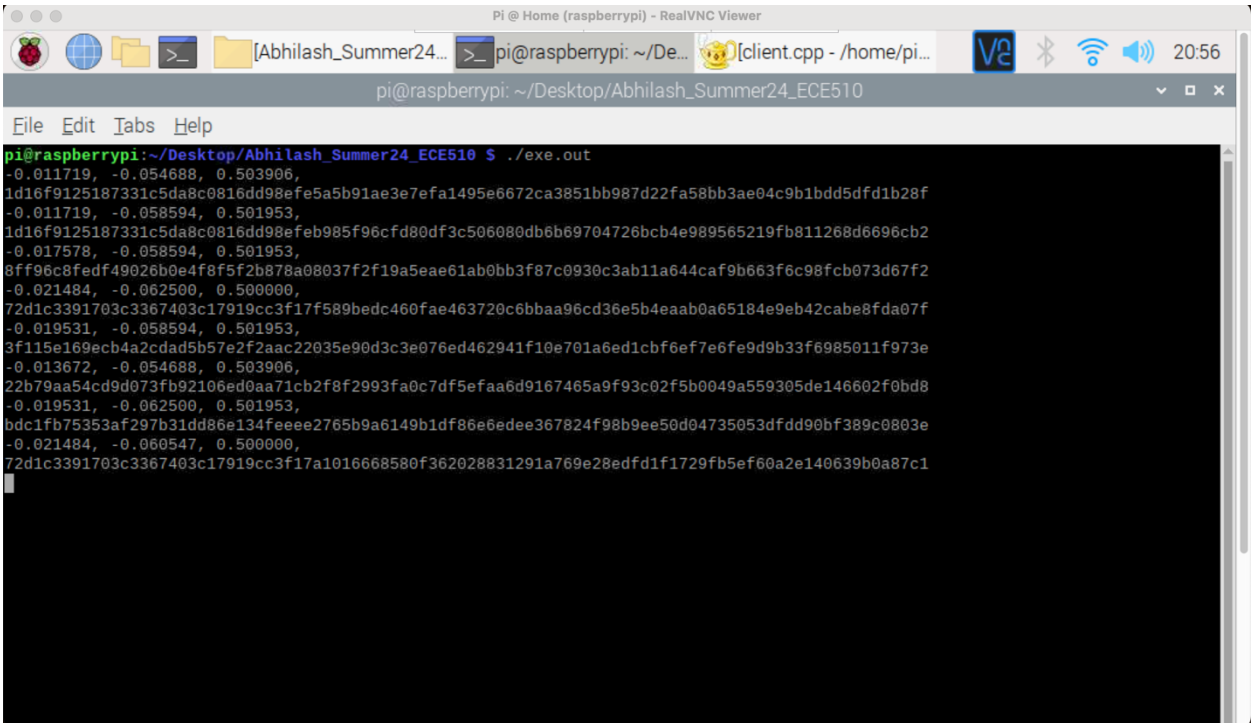

Figure 5: Client begins to transmit data and the same visible on the client side on Terminal.

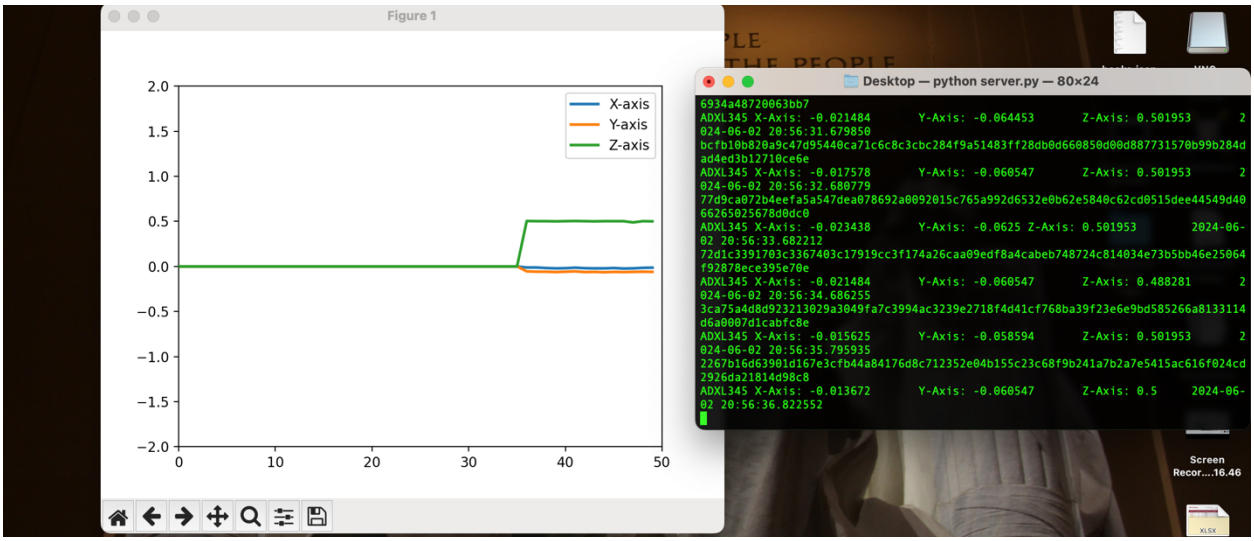

Figure 6: Server Capturing information from Client (1/2).

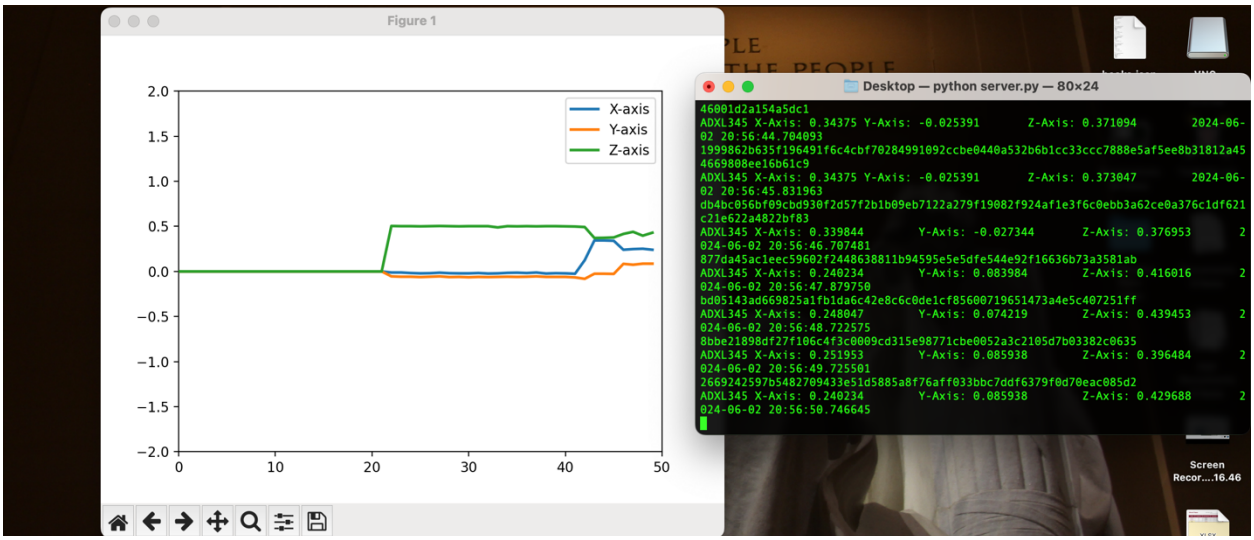

Figure 7: Server Capturing information from Client (2/2).

A. Discussion

1. Differentiation of Slave devices in I²C and SPI Bus

SPI (Serial Peripheral Interface) and I2C (Inter-Integrated Circuit) protocols both use unique identifiers that the master device uses to distinguish amongst slave devices. The two protocols do not, however, use these identifiers in the same way.

Every slave device has a unique address allocated to it in the I2C protocol. The master device delivers the desired slave device's 7-bit address and starts a START condition on the bus when it wants to speak with that slave device. is followed by a little piece that indicates the operation (write or read). By bringing the data line low, the slave device that has the matching address notifies the master (ACK bit). The master can send a repeated START to try again or a STOP condition to halt the transmission if no device acknowledges.

However, unlike I2C, the SPI protocol does not employ an addressing system. Rather, every slave device is assigned its own Slave Select (SS) line. By lowering its SS line, the master device chooses which slave to talk to. Every slave in a system with several slaves would have a separate SS line

leading back to the master. This can be detrimental in pin-constrained systems since it means that as the number of slaves rises, so does the number of GPIO pins needed on the master.

2. **Behavior of ADXL345 sensor with 0x2F in Register 0x31**

Three-axis accelerometer with numerous configurable registers is the ADXL345. The DATA_FORMAT register, located at 0x31, regulates how data is displayed. It configures the device's self-test mode, the full-scale range of the accelerometer readings, and the data justification. The accelerometer is in FULL_RES mode with a range of ±16g (bits D3 and D1 are set) and the data is left-justified with sign extension (bit D2 is set) if the value 0x2F is detected in register 0x31. Additionally, the self-test feature is activated (bit D7 is set), which simulates a fixed acceleration by applying a known electrostatic force to the sensor. This is done to make sure the sensor is operating properly.

3. **Delay in Data output from Physical Accelerometer to Server Side**

There are several reasons for the lag that occurs between the accelerometer's actual movement and the server-side output of the accelerometer data that is received. These consist of the accelerometer's data conversion time, the microcontroller's processing time, the transmission time via the Wi-Fi network, the processing time on the server, and the transmission time over the I2C or SPI bus.

Numerous actions can be taken to enhance the server-side real-time data responsiveness. Reducing the precision or sample rate, or selecting an accelerometer with a faster conversion time, will shorten the accelerometer's data conversion time. One can decrease the amount of data being communicated or increase the speed of the bus to shorten the transmission time over the SPI or I2C bus. Code optimization or the use of a faster microprocessor are two ways to shorten the microcontroller's processing time. Reducing the quantity of data being communicated or utilizing

a faster network can both shorten the Wi-Fi transmission time. Code optimization or the use of a faster server can both shorten the server's processing time.

4. **Maximum Sampling Rate for the Server (Secondary Device)**

The slowest connection in the sensor-to-server network determines the maximum sample rate for the PC server. This covers the sensor's sampling rate, the I2C bus bandwidth, the Wi-Fi bandwidth, and the data encoding used during transmission.

The system must be set up to optimize the sensor's sampling rate, the I2C bus bandwidth, the Wi-Fi bandwidth, and the data encoding efficiency to reach the theoretical maximum sampling rate on the server. This could entail utilizing a fast I2C bus and Wi-Fi network, selecting a sensor with a high maximum sampling rate, and sending the data as raw floating-point data. It's crucial to remember that raising the sampling rate might also result in higher power consumption, which may be problematic for battery-operated devices.

References

[1] Experiment 01 Lab Manual