Software Overview

Year: 2019 Semester: Fall Team: 10 Project: Gesture controlled smart home

Creation Date: 8/26/2019 Last Modified:

Member 1: Wan Hsuan Lo Email: lo51@purdue.edu

Member 2: Timothy Yu-Jay Huang Email: huang983@purdue.edu

Member 3: Tsung Lin Hsia Email: thsia@purdue.edu

Member 4: Xi Wu Email: wu999@purdue.edu

Assignment Evaluation:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item** | **Score (0-5)** | **Weight** | **Points** | **Notes** |
| **Assignment-Specific Items** | | | | |
| **Software Overview** | 5 | x2 | 10 | Good but more than software it is leaning towards communication protocols. |
| **Description of Algorithms** | 4.5 | x2 | 9 | Good description. |
| **Description of Data Structures** | 4 | x2 | 8 | The TCP/IP data needs more detail. |
| **Program Flowcharts** | 5 | x3 | 15 | Needs to handle wake up signal |
| **State Machine Diagrams** | 5 | x3 | 15 | Looks good |
| **Writing-Specific Items** | | | | |
| **Spelling and Grammar** | 4.5 | x2 | 9 | Minor errors here and there. |
| **Formatting and Citations** | 4.5 | x1 | 4.5 | Use IEEE citation format |
| **Figures and Graphs** | 4.5 | x2 | 9 | Figures can be placed properly and made of standard size. |
| **Technical Writing Style** | 5 | x3 | 15 | Good |
| **Total Score** | 94.5 | | |  |

5: Excellent 4: Good 3: Acceptable 2: Poor 1: Very Poor 0: Not attempted

General Comments:

*Relevant overall comments about the paper will be included here*

Inline

1.0 Software Overview

Our design requires three main firmware interfaces: An I2C interface for receiving gesture sensor data, an SPI interface for communicating with the LCD display, and a UART interface for WiFi module.

The flow of the overall firmware operation can be seen from the state machine attached to this document below. Firstly, the gesture sensor reads the user’s gesture, and displays the current status on the LCD. If a valid gesture is detected, the sensor will send the gesture data to the microcontroller. The microcontroller decodes the I2C data and delivers the wanted instruction to the WiFi module. If the WiFi module successfully receives the command from the microcontroller, it sends the command through a UART interface to IFTTT and let IFTTT invoke the designated smart home device and send the command. IFTTT is a web-based service that helps creates a simple chain between a developer’s chosen device to smart home devices. Once the command is successfully delivered to the designated smart home device, it sends a confirmation message back to the WiFi module, letting it know that the designated smart home device has received the command.

Everytime a certain stage is reached, the LCD display would update the user on the status using SPI. The display would show the user what instruction is sent; whether the instruction is successfully delivered; and if not in which step an error has occurred.

Once the gesture sensor receives a valid gesture, it needs to notify the microcontroller through the I2C communication protocol. Therefore an I2C interface is necessary.

As for the functionality of displaying device status on the LCD display, we would implement an SPI interface between the microcontroller (master) and the display (slave). Every time the microcontroller moves into a next state, it would send a MOSI signal to the display with the new status. The display would then reply with MISO on whether the update was successful.

The WiFi module utilizes a UART interface to communicate with the microcontroller. It creates a WiFi connection and acts as a buffer and status reporter between the IFTTT server and our microcontroller. On the other hand, to establish a connection between the WiFi module and IFTTT, the device would need software implementation of a device API and a shim between the two different APIs.

2.0 Description of Algorithms

The main algorithms that need to be implemented or designed are for gesture trigger function and connection between WiFi module and IFTTT.

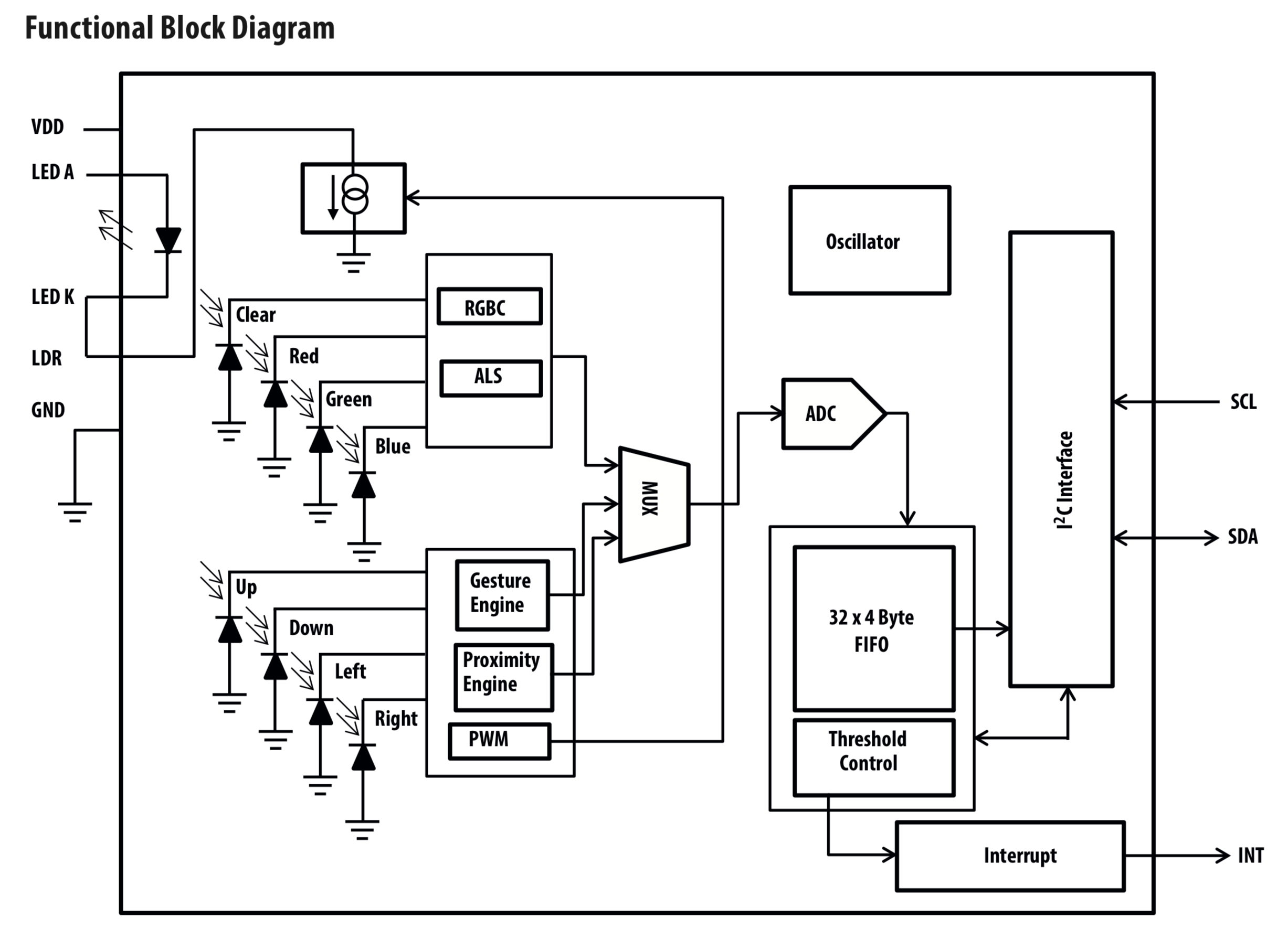


Figure 1. Gesture Sensor Block Diagram[1]

For the implementation of the gesture trigger function, first of all, the interrupt of the sensor would be triggered internally to bring the sensor out of sleep mode every time the device resets. Then we would send a read command to access the 32\*4 byte FIFO registers as seen above in the block diagram to see if any gestures has been triggered. To see which gesture has been triggered, we consider these following registers as most helpful.

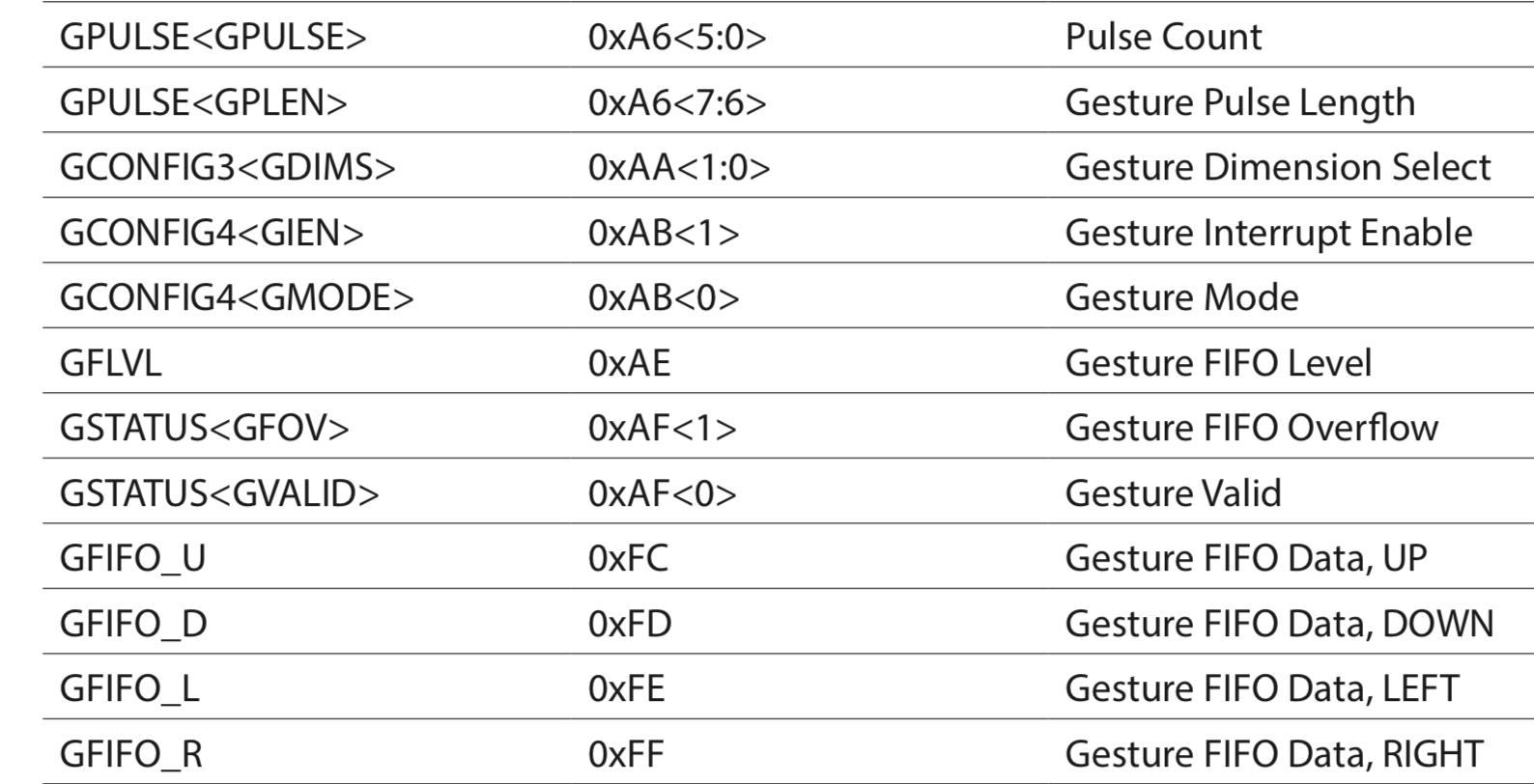


Figure 2. FIFO registers [1]

Registers 0xFC to 0xFF contain simple detection of up, down, left, right gestures. With the combination of gesture pulse count, pulse length and FIFO level, we could decide if other more complicated gestures have been made, such as waving multiple times, holding hand still in front of sensor, etc. The specific calibration of which values to expect would require testing on the actual sensor in a setup environment.

In order to connect ESP8266 with IFTTT services, we will be using the official ESP8266 WiFi and web server libraries to establish network connection [7] [8]. Once the WiFi module is configured, we will define our own IFTTT trigger events, for example, “turn on the lights”, “turn off the lights”, “volume up”, etc. Whenever a command is passed to the WiFi module it sends the corresponding IFTTT trigger event to the IFTTT applet. If the event matches with the event defined in IFTTT applet, IFTTT applet will execute the command. For instance, if the microcontroller sends the command “turn off the lights” for Philip Hue to the WiFi module, the WiFi module will first find the corresponding IFTTT trigger event to that command and send it as a packet to the IFTTT applet we create, and then the applet takes the command and sends it to the designated smart home device.

3.0 Description of Data Structures

Sensor:

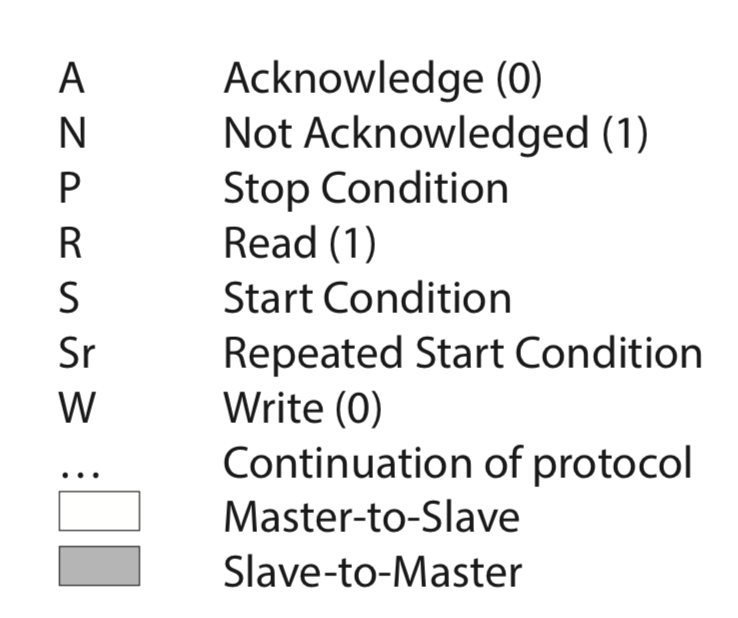
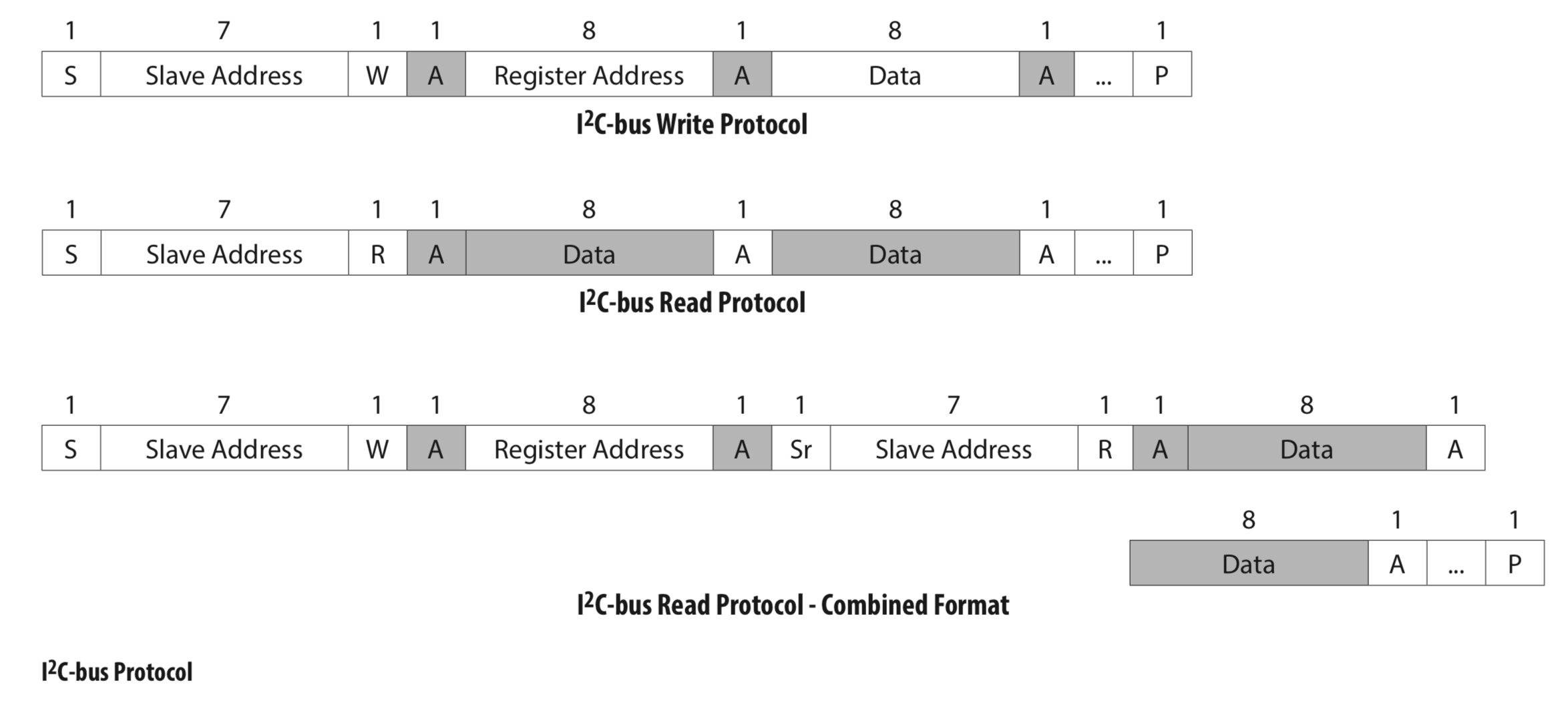


Figure 3. I2C Bus Protocol[1]

The data structure for the gesture sensor is 7-bit addressing I2C protocol. For example, to initiate a read register command, the data sent would be 1 bit condition, 5 bit register address, and 1 bit opcode for read; the sensor slave would then return the data requested in a series of bytes.

WiFi Module:

Read/Write Buffer: DMA will directly send packets received and sent by the ESP8266 SDIO to corresponding memories. The ESP8266 will define the linked list registration structure and buffers.

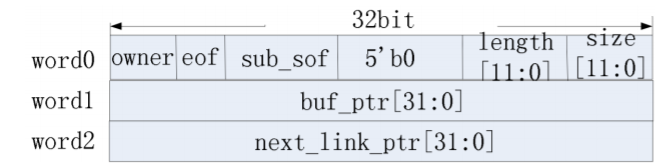


Figure 4.The linked list registration structure[5]

As one can see from the figure above, the packets consist of multiple words, with the first word describing the packet, such as the size to be expected and the length of the data.

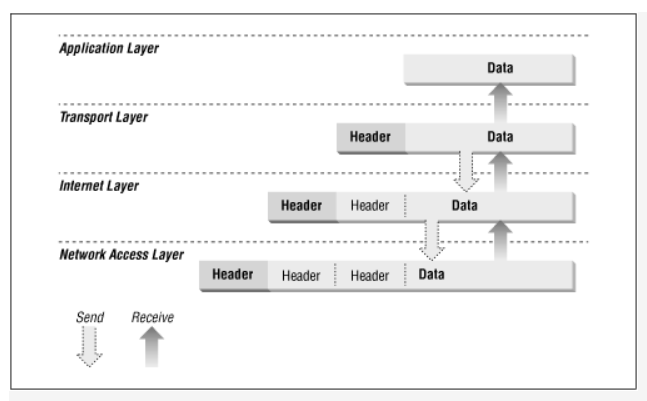


Figure 5. TCP/IP Protocol Architecture[6]

4.0 Sources Cited:

[1]Avago Technologies (2013, November 08) APDS-9960 Gesture Sensor Datasheet: <https://cdn.sparkfun.com/assets/learn_tutorials/3/2/1/Avago-APDS-9960-datasheet.pdf>

[2] P. Pieter (2017, August 03) *A Beginner's Guide to the ESP8266: Wi-Fi*

<https://tttapa.github.io/ESP8266/Chap05%20-%20Network%20Protocols.html>

[5] Espressif IoT Team(2019, August) ESP8266 Technical Reference

<https://www.espressif.com/sites/default/files/documentation/esp8266-technical_reference_en.pdf>

[6] Robert Bruce Thompson, Craig Hunt (1998, August) TCP/IP Network Administration

<https://www.oreilly.com/library/view/windows-nt-tcpip/1565923774/>

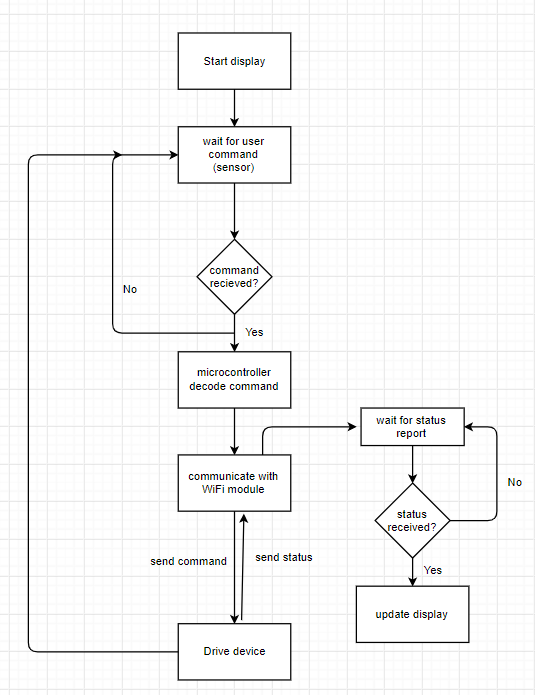
[7] (2019, September 04) *ESP8266WiFi*

<https://github.com/esp8266/Arduino/blob/master/libraries/ESP8266WiFi/src/ESP8266WiFi.h>

[8] (2019, August 14) *ESP8266WebServer*

<https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WebServer/src>

Appendix 1: Program Flowcharts



Appendix 2: State Machine Diagrams

