Utilizing TI's CC3220SF wireless microcontroller Launchpad development kit, the same has been configured as an HTTP Server connected to the Embedded Lab Wi-Fi (Embedded\_Lab\_EXT) to read accelerometer data for X and Y axes wirelessly through the python program given below. With the board i.e., HTTP Server also acquiring accelerometer values for X and Y axes through I2C Communication, the same have been requested from the board through calling the HTTP GET method API’s defined in the program ‘link\_local\_task.c’. With the values obtained for both axes, being encoded as a bytes object, the same is decoded into a string which is then split to obtain the values and cast into integers to be provided to the variables x\_new and y\_new. With the variables x\_new and y\_new being parsed to the patch.center () method, the values are utilized to determine and undertake movement of ball over the screen. With the ball on execution of program being set at coorodinates (0, 0), tilting of the board in any direction results corresponds to movement of ball over the screen undertaken through the API of the HTTP GET method which is

Utilizing TI’s CC3220SF board, the same has been configured as an HTTP server and connected to the Embedded Lab Wi-Fi (Embedded\_Lab\_EXT). With the board also obtaining accelerometer values along the X and Y axes through I2C Communication, the very values are accessed wirelessly through the python program below through HTTP GET method (API’s of which are defined in the program ‘link\_local\_task.c’). With values obtained being encoded in the bytes format, the same is decoded into strings which then been split, the values cast into integer to be assigned to variables x\_new and y\_new for X and Y axes respectively. With the variables being assigned to the patch.center () method, they facilitate the movement of ball over the screen which is set at coordinates (0, 0) upon the program being executed, in accordance with the direction of tilt of the board. With the HTTP Get Request being sent to the board through the program continuously, which is displayed over the TERA Term through Serial communication, tilting the board in any direction will result in movement of ball over the screen in accordance with the accelerometer values along the X and Y axes.

The following set of specifications were derived from lab requirements:

1. Start with how GET/POST works in the HTTP protocol and specifically how to pass a parameter with a GET.
2. After connecting to the embedded lab WIFI, it is time to learn about the I2C protocol. First, understand how its addressing works and tries to read the integrated temperature sensor.
3. Having grasped I2C protocol and its addressing techniques and having noted the default save address of Accelerometer (in hex) to be 0x18, the ‘out\_of\_box\_CC3220SF\_LAUNCHXL\_tirtos\_ccs’ was next imported from Resource Explorer and complied. Launching Tera Term application to view UART output data (mainly IP of the board), the program was run with the Baud rate set to 115200, Data Bits set to 8, a single stop bit and no parity..
4. Connecting laptop to the board Wi-Fi, browse to IP address of the board 10.123.45.1 from the browser to add profile for the Wi-Fi of Embedded Lab as follows:

SSID: Embedded\_Lab\_Ext

Security Type: WPA/WPA2

Security Key: embedded

Profile Priority: 15

1. Restarting the program through the CCS, the board was now connected to the Embedded Lab Wi-Fi and connfugured as an HTTP Server through the program ‘out\_of\_box\_CC3220SF\_LAUNCHXL\_tirtos\_ccs’ with IP of 10.0.0.61, responding with accelerometer data along the X and Y axes for the following URL’s

{X-Axis}: [http://<BOARD\_IP>/sensor?axisx]

{Y-Axis}: ["http://<BOARD\_IP>/sensor?axisy]

An I2C Transaction is also undertaken by the board to acquire the accelerometer values along X and Y axes with I2C0 and a communication speed of 400 KHz. I2C address for X and Y axes is 0x02 and 0x04 respectively.

1. With the data obtained from the board being encoded as a bytes object, the same will be decoded into a string which is then split to obtain the value and cast into an integer to be assigned to variables x\_new and y\_new for X and Y axes respectively.
2. With the variables being parsed to the parse.center () method, the accelerometer values obtained from the board wirelessly through HTTP GET method API will be used to determine and undertake movement of ball which is initially centred at coordinates (0, 0) upon execution, in accordance with the tilt of the board as the latter will change accelerometer values which in turn will cause ball movement over the screen.
3. Additionally, the HTTP Get request received and responded to by the board for X and Y axes accelerometer data can be viewed through the UART over the Tera term terminal.

The following steps were undertaken to make CC3220SF read Temperature Sensor values:

The Inter Integrated Circuit (I2C) supports bidirectional data transfer in half duplex mode through a synchronous bus and 2 channels viz. SDA and SCL. With the master (the processor, in this case) initiating the start condition over the falling edge of SDA line while SCL is high, it then sends the address of device it wishes to communicate with over the SDA followed by the Read /write bit. While 0 indicates that the master will write, 1 indicates that the master will read from the device. Initiating communication in the aftermath of receiving ACK from the slave device which is done by the slave holding the data line low, the master will read data from / write data to the slave device on completion of which it sends the stop condition by making the SDA line high which SCL is held high. With the digital temperature sensor of the CC3220SF board being I2C Compatible, following steps were undertaken to read temperature values from it.

1. Referring to the User's guide of the CC3220SF board, default slave address of the Temperature sensor was obtained (0x41).
2. Initially, opening up the TMP006Driver, the manufacturer and version of the board is obtained upon verification of which i.e. upon verifying that board is Texas Instruments CC3220SF, the configuration register value will be obtained.
3. Obtaining the Sensor voltage register value in the aftermath (i.e. at address 0x0), the ambient temperature register value will be acquired from the address 0x1 through the read from I2C API (to get the required number of bytes)
4. Applying Format Conversion to value obtained, the same is then converted into Farenheits and assigned to variable ‘temperatureVal’ if the value has been successfully acquired.
5. With the variable holding the current temperature value, the same is appended and sent with the payload as response to the HTTP GET request received while displaying the request reieved and values sent over thte Tera Term Termainal through UART.

For obtaining the accelerometer values for the X and Y axes, the following steps are undertaken:

1. Referring to the User's guide of the CC3220SF board, default slave address of the Accelerometer was obtained (0x18).
2. Initially setting the register to be written to, the read from I2C API is invoked to acquire the required number of bytes from the slave address (i.e. 0x18).
3. Initializing the BMA222 accelerometer, in the aftermath the chip ID number will be obtained through the BMA222Open () function followed by the accelerometer data readings through pointers \*psAccX and \*psAccY for X and Y axes which point towards the raw AccX value and raw AccY value stores respectively.
4. Reading the LSB and MSB blocks through BlockRead () function from the store for both axes in the BMA222ReadNew () method, the board will periodically check for new values X and Y axes, in the event of which the new values will be acquired to replace the old acquired values which shall assigned to variables xval and yval after range conversion.
5. With the variables having the accelerometer values along X and Y axes, the variable values (either X or Y axis value) are appended and sent with the payload as response to the received HTTP GET request from the below python code while displaying request received and values sent over the Tera Term Terminal through UART.