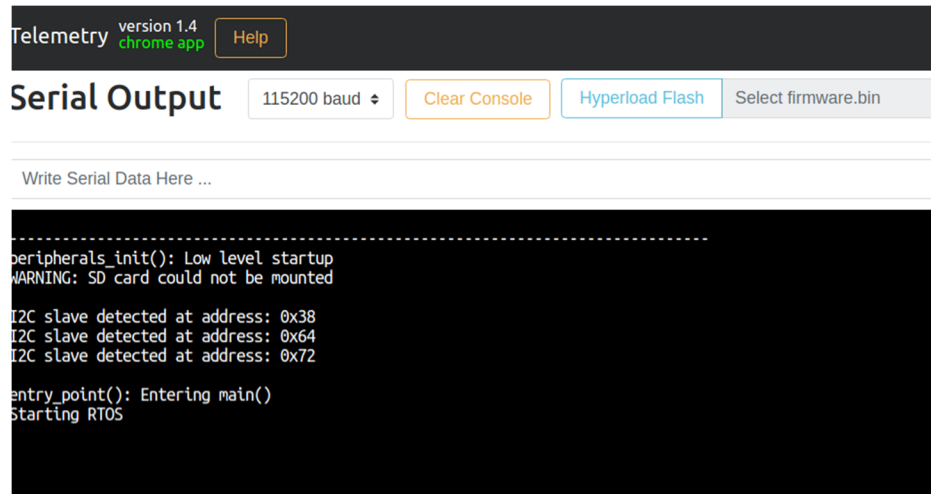


## LAB 10: I2C

### PART 1:

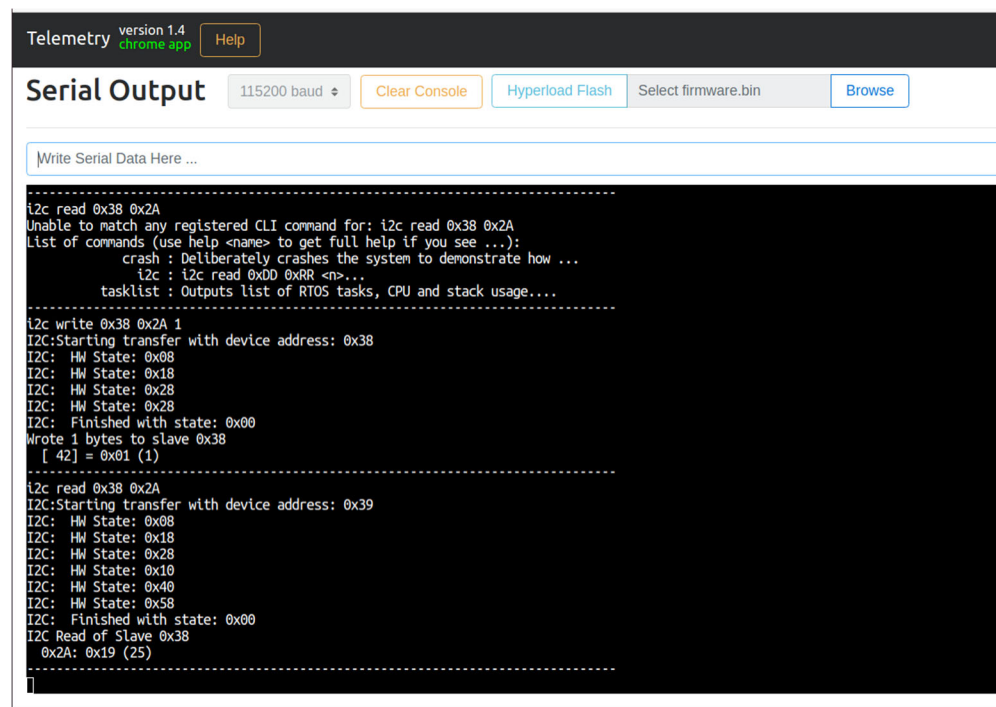
- a) The following figure shows slave addresses of I2C read from telemetry.



The screenshot shows the Telemetry interface with the Serial Output window open. The baud rate is set to 115200. The console output displays the following text:

```
-----  
peripherals_init(): Low level startup  
WARNING: SD card could not be mounted  
  
I2C slave detected at address: 0x38  
I2C slave detected at address: 0x64  
I2C slave detected at address: 0x72  
  
entry_point(): Entering main()  
Starting RTOS
```

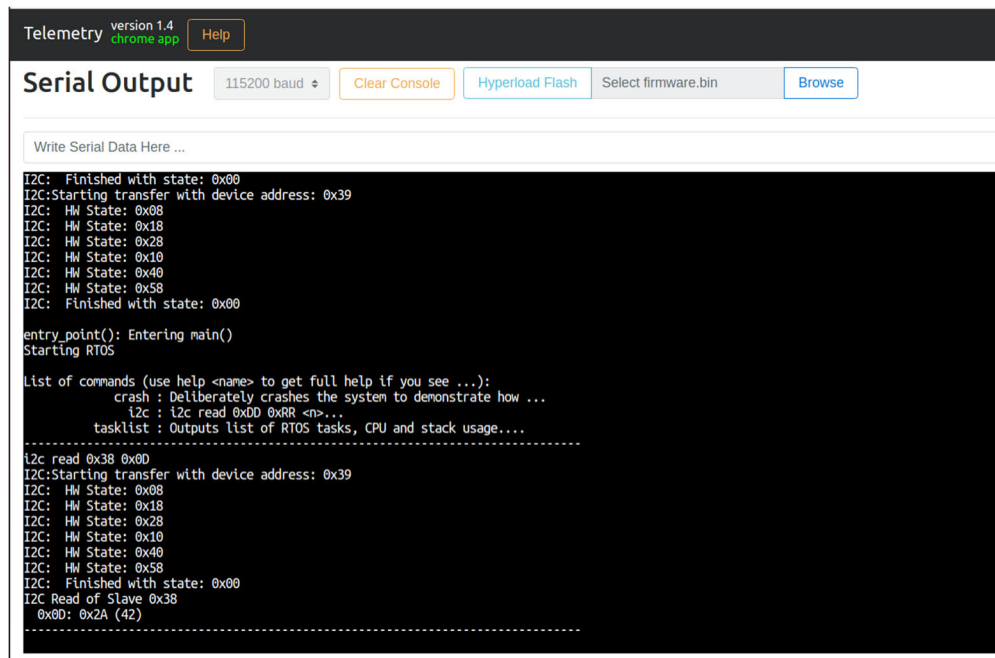
- b) The attached file I2C\_Enable.txt with this code shows the printf statements after debugging was enabled
- c) The following figure is output after writing values into **CTRL\_REG1** (0x2A) to activate the sensor



The screenshot shows the Telemetry interface with the Serial Output window open. The baud rate is set to 115200. The console output displays the following text:

```
-----  
i2c read 0x38 0x2A  
Unable to match any registered CLI command for: i2c read 0x38 0x2A  
List of commands (use help <name> to get full help if you see ...):  
  crash : Deliberately crashes the system to demonstrate how ...  
  i2c : i2c read 0x00 0xRR <n>...  
  tasklist : Outputs list of RTOS tasks, CPU and stack usage...  
-----  
i2c write 0x38 0x2A 1  
I2C:Starting transfer with device address: 0x38  
I2C: HW State: 0x08  
I2C: HW State: 0x18  
I2C: HW State: 0x28  
I2C: HW State: 0x28  
I2C: Finished with state: 0x00  
Wrote 1 bytes to slave 0x38  
  [ 42] = 0x01 (1)  
-----  
i2c read 0x38 0x2A  
I2C:Starting transfer with device address: 0x39  
I2C: HW State: 0x08  
I2C: HW State: 0x18  
I2C: HW State: 0x28  
I2C: HW State: 0x10  
I2C: HW State: 0x40  
I2C: HW State: 0x58  
I2C: Finished with state: 0x00  
I2C Read of Slave 0x38  
  0x2A: 0x19 (25)  
-----  
[
```

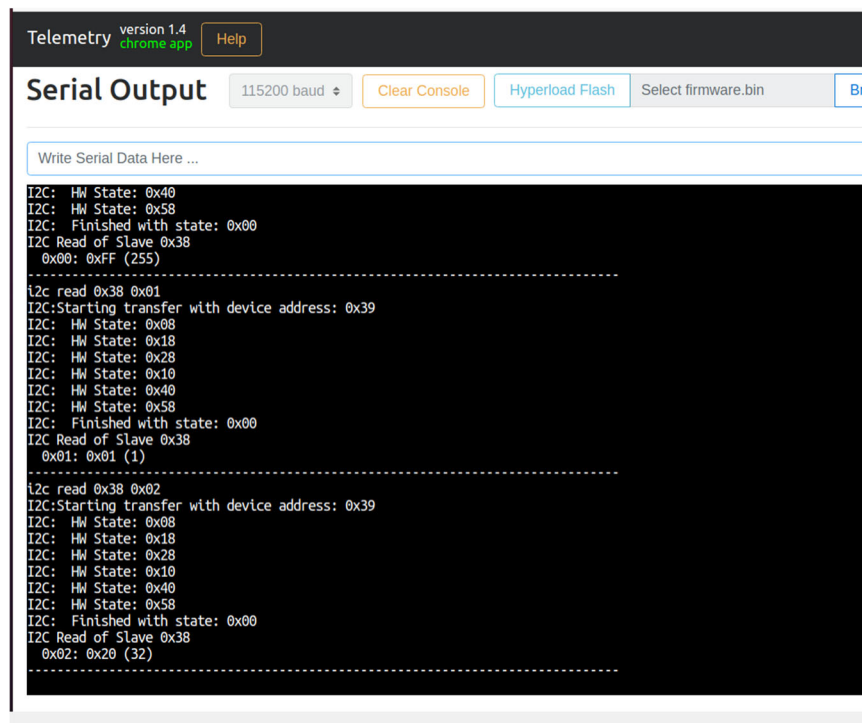
- d) The following figure is output after writing values into **WHO\_AM\_I** (0x0D) register to activate the sensor:



The screenshot shows the Telemetry interface with a 'Serial Output' tab. The output text is as follows:

```
Telemetry version 1.4 chrome app Help
Serial Output 115200 baud Clear Console Hyperload Flash Select firmware.bin Browse
Write Serial Data Here ...
I2C: Finished with state: 0x00
I2C: Starting transfer with device address: 0x39
I2C: HW State: 0x08
I2C: HW State: 0x18
I2C: HW State: 0x28
I2C: HW State: 0x10
I2C: HW State: 0x40
I2C: HW State: 0x58
I2C: Finished with state: 0x00
entry_point(): Entering main()
Starting RTOS
List of commands (use help <name> to get full help if you see ...):
  crash : Deliberately crashes the system to demonstrate how ...
  i2c : i2c read 0x00 0xRR <n>...
  tasklist : Outputs list of RTOS tasks, CPU and stack usage....
-----
i2c read 0x38 0x00
I2C: Starting transfer with device address: 0x39
I2C: HW State: 0x08
I2C: HW State: 0x18
I2C: HW State: 0x28
I2C: HW State: 0x10
I2C: HW State: 0x40
I2C: HW State: 0x58
I2C: Finished with state: 0x00
I2C Read of Slave 0x38
  0x0D: 0x2A (42)
-----
```

- e) The following 3 screenshots is the values read from XYZ of accelerometer:



The screenshot shows the Telemetry interface with a 'Serial Output' tab. The output text is as follows:

```
Telemetry version 1.4 chrome app Help
Serial Output 115200 baud Clear Console Hyperload Flash Select firmware.bin Br
Write Serial Data Here ...
I2C: HW State: 0x40
I2C: HW State: 0x58
I2C: Finished with state: 0x00
I2C Read of Slave 0x38
  0x00: 0xFF (255)
-----
i2c read 0x38 0x01
I2C: Starting transfer with device address: 0x39
I2C: HW State: 0x08
I2C: HW State: 0x18
I2C: HW State: 0x28
I2C: HW State: 0x10
I2C: HW State: 0x40
I2C: HW State: 0x58
I2C: Finished with state: 0x00
I2C Read of Slave 0x38
  0x01: 0x01 (1)
-----
i2c read 0x38 0x02
I2C: Starting transfer with device address: 0x39
I2C: HW State: 0x08
I2C: HW State: 0x18
I2C: HW State: 0x28
I2C: HW State: 0x10
I2C: HW State: 0x40
I2C: HW State: 0x58
I2C: Finished with state: 0x00
I2C Read of Slave 0x38
  0x02: 0x20 (32)
-----
```

Figure: X-Axis

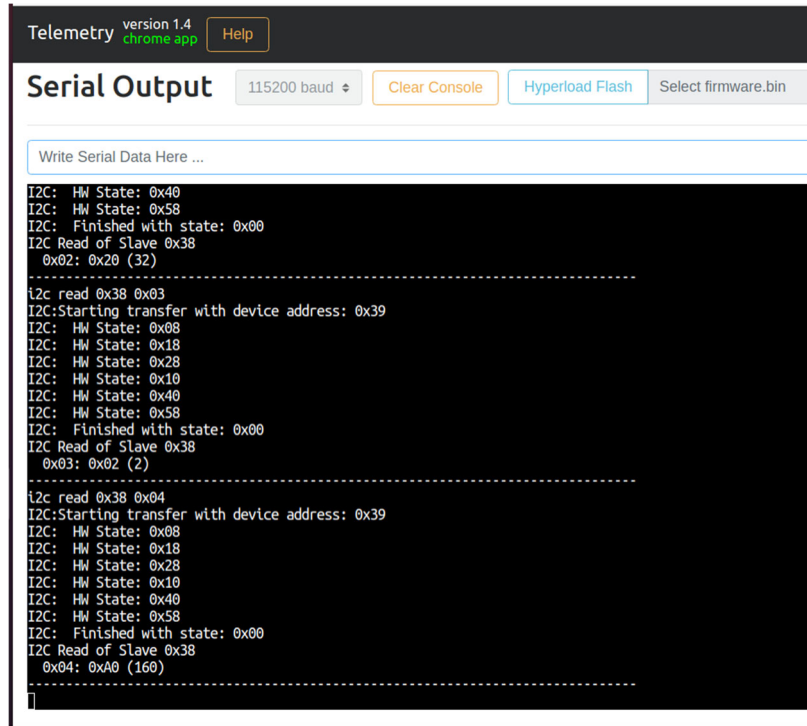


Figure: Y-Axis

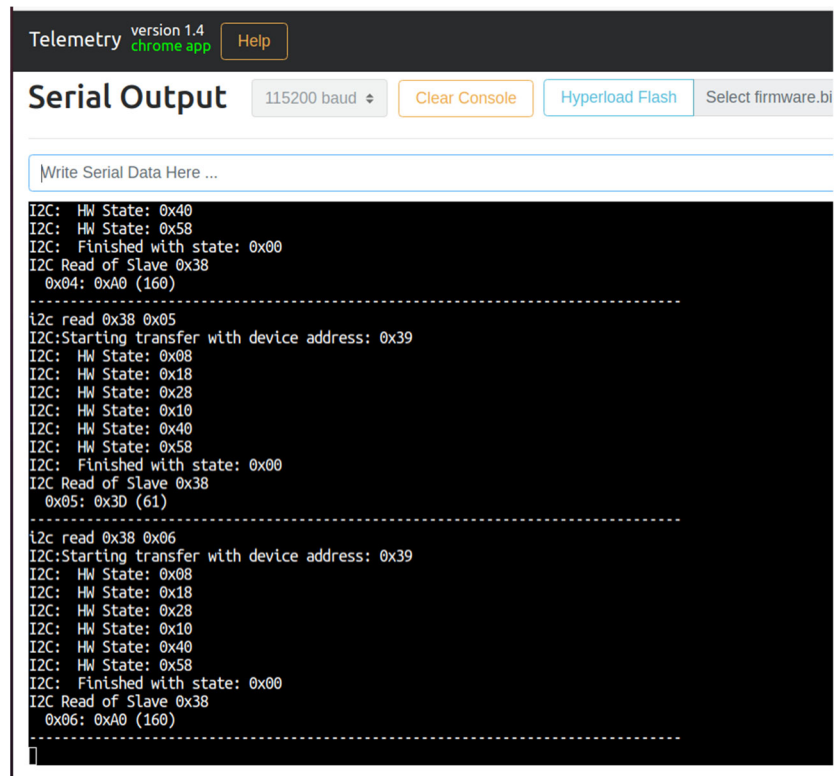


Figure: Y-Axis

## PART 2: State Diagram when Slave is Transmitting and Receiving

The attached excel sheet in repository gives a detailed analysis of what each status code represents. Below is just an example when reading and writing a memory location. The table explains the meaning of each state represented by the state diagram.

The following represents state diagram and table which explains the steps when slave is considered as receiver and Master needs to write data into a memory location.

Writing into the Memory Location								
	START	SLA+W	ACK	Memory Location	ACK	DATA	ACK	STOP
SLAVE STATE			0x60	0x80		0x80		0xA0
							Repeated fro N Bytes	
				Repeated for N different addresses				

State Diagram 1

HW State	Why you got here?	Action to take
Writing into the Memory Location: Slave Read		
Initially a START bit is sent from the Master side followed by SLA+W to indicate the slave address and write operation into the Slave.		
0x60	Slave receives its own address along with the W bit	ACK is sent by slave. Then wait for next data byte from Master which would signify the register address.
0x80	Slave receives the data from Master which corresponds to the memory location to be written.	Slave in return send ACK bit and waits for the data to be written
0x80	Slave receives the data from Master which corresponds to the data which should be written into the memory location received from above.	Slave in return send ACK bit and waits for the next data to be written
0xA0	Master sends a STOP bit to stop this transaction	There is no acknowledgment sent back to the Master. SI bit is cleared.

Table 1

The following represents state diagram and table which explains the steps when slave is considered as transmitter and Master needs to read data from a memory location.

Reading from the Memory Location													
	START	SLA+W	ACK	Memory Location	ACK	Repeat START	SLA+R	ACK	DATA	ACK	DATA	NACK	STOP
SLAVE STATE		0x60		0x80			0xA8		0xB8		0xC0		0xC8
									Repeated fro N Bytes		Last Byte		

State Diagram 2

HW State	Why you got here?	Action to take
Slave Write: Reading from Memory Location		
Initially START bit is sent from Master followed by SLA+W and in return the slave has to acknowledge this bit. The Master has to write the first data byte., then send a REPEAT START bit and then send SLA+R to start transaction.		
0xA8	Slave receives its address and Read bit.	ACK is sent by slave.
0xB8	As soon as read bit is received Slave will send the ACK and transmit first data byte.	Master will send ACK in return and wait for next byte. Slave needs to verify if ACK is received or NACK. If ACK is received it is ready to send the next byte and stay in this state.
0xB8	This state is repeated for N bytes. Each time ACK is received from Masters side	Master will send ACK in return and wait for next byte. Slave needs to verify if ACK is received or NACK. If ACK is received it is ready to send the next byte and stay in this state.
0xC0	The data byte is transmitted from Slave and Master send NACK to stop the transaction	Slave waits for reception of stop bit.
0xC8	Last Data byte is transmitted and NACK is received.	Slave stops the transaction and clear SI.

Table 2