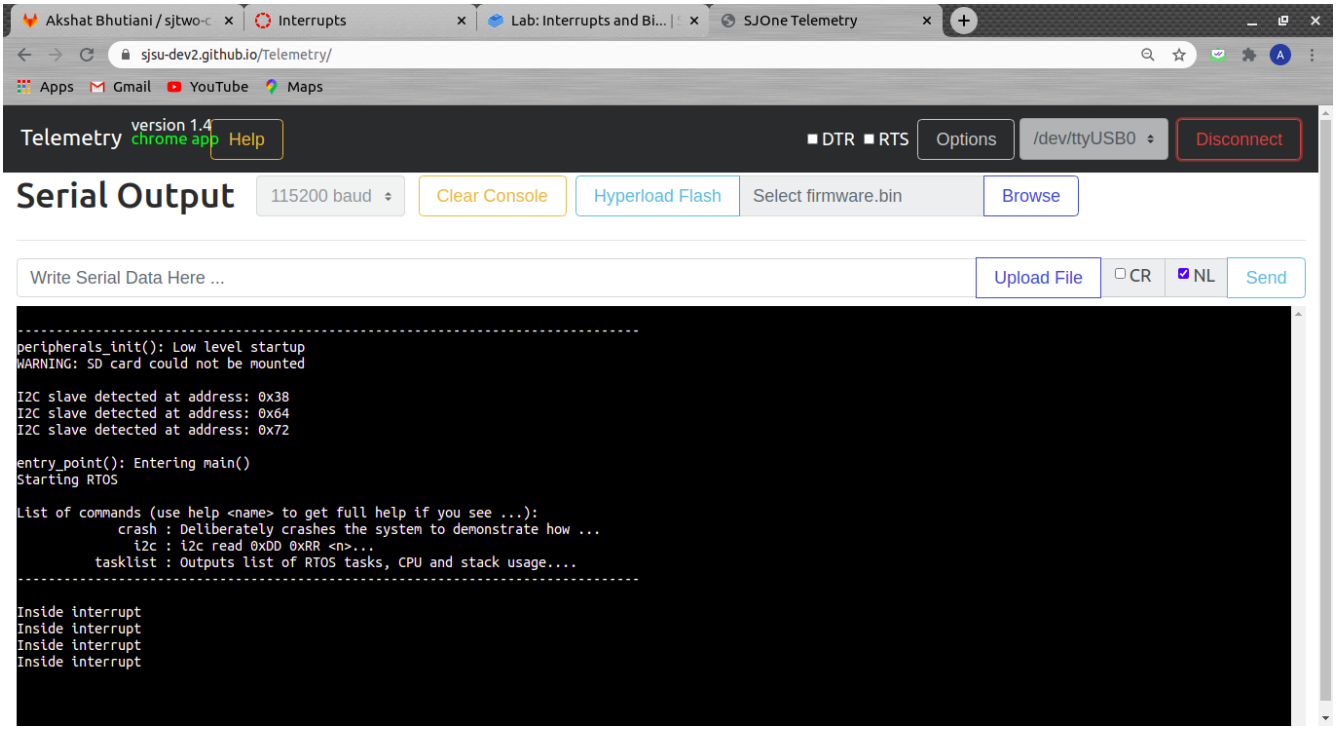
Name: Akshat Bhutiani
Student ID: 014538238

Part 0:

Code -

```
void gpio_interrupt(void) {
  LPC_GPIOINT->IO0IntClr |= (1U << 30);  // Clear the interrupt
  fprintf(stderr, "\nInside interrupt"); // print inside the interrupt
}

int main(void) {
  // create_blinky_tasks();
  create_uart_task();

  // Part 0:
  const uint32_t sw1 = (1 << 30);
  LPC_GPIO0->DIR &= ~sw1;        // set switch as input;
  LPC_IOCON->P0_29 |= (1 << 3);  // enable pulldown registers
  LPC_GPIOINT->IO0IntEnF |= sw1; // configure falling edge interrupt

  const uint32_t led18 = (1U << 18);
  LPC_GPIO1 -> DIR |= led18;

 while (1) {
    delay__ms(500);
    LPC_GPIO1 -> SET = led18;
    delay__ms(500);
    LPC_GPIO1 -> CLR = led18;
    delay__ms(500);
  }

  NVIC_EnableIRQ(GPIO_IRQn);

  puts("Starting RTOS");
  vTaskStartScheduler(); // This function never returns unless RTOS scheduler runs out of memory and
fails

  return 0;
}
```

Output -



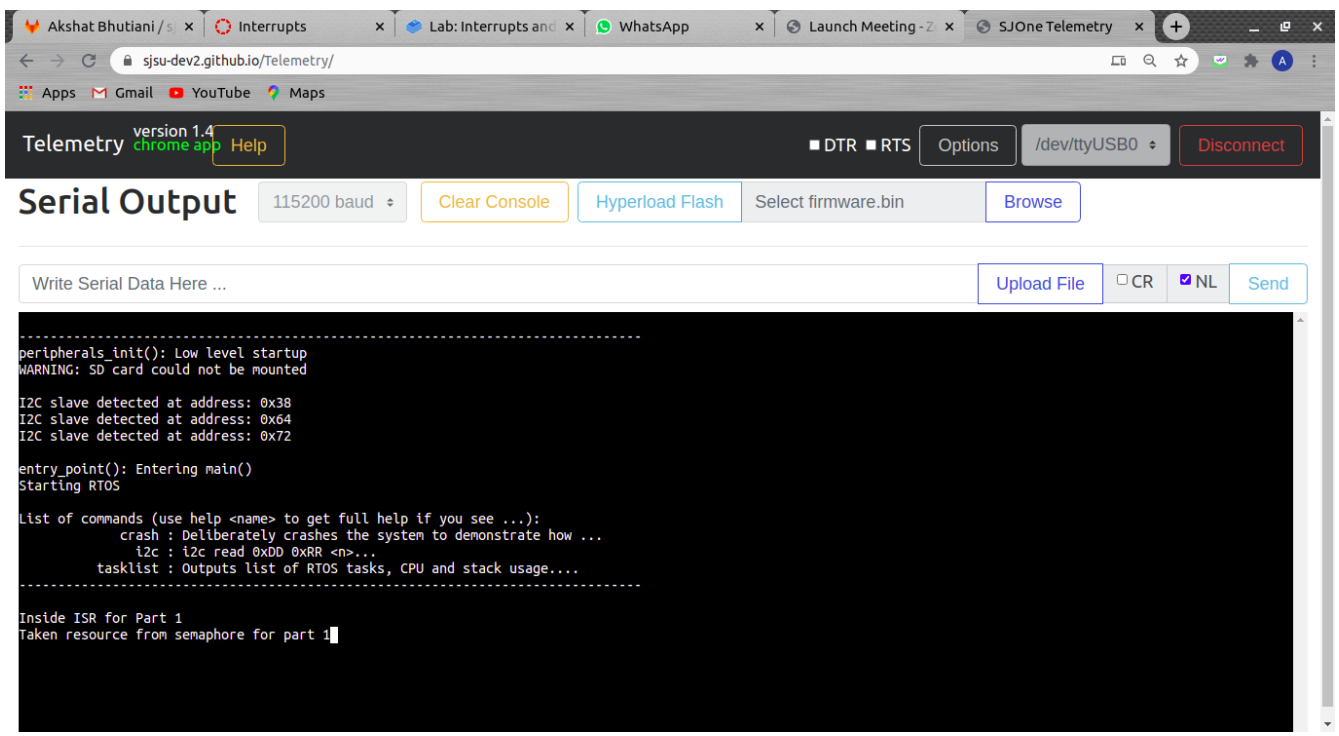
Part 1:

Without call backs -

```
static SemaphoreHandle_t switch_pressed_signal;

void gpio_interrupt(void) {
  LPC_GPIOINT->IO0IntClr |= (1U << 30); // Clear the interrupt
  fprintf(stderr, "\nInside ISR for Part 1");
  xSemaphoreGiveFromISR(switch_pressed_signal, NULL);
}

void sleep_on_sem_task(void *params) {
  while (1) {
    if (xSemaphoreTake(switch_pressed_signal, 1000)) {
      fprintf(stderr, "\nTaken resource from semaphore for part 1");
    }
  }
}

int main(void) {
  // create_blinky_tasks();
  create_uart_task();
```

```
  // Part 1:
  switch_pressed_signal = xSemaphoreCreateBinary(); // create binary semaphore

  const uint32_t sw1 = (1 << 30);
  LPC_GPIO0->DIR &= ~sw1;        // set switch as input;
  LPC_IOCON->P0_29 |= (1 << 3);  // enable pulldown registers
  LPC_GPIOINT->IO0IntEnF |= sw1; // configure falling edge interrupt

  const uint32_t led18 = (1U << 18);
  LPC_GPIO1->DIR |= led18;

  NVIC_EnableIRQ(GPIO_IRQn);

  xTaskCreate(sleep_on_sem_task, "sem", (512U * 4) / sizeof(void *), NULL, PRIORITY_LOW,
NULL);
  puts("Starting RTOS");
  vTaskStartScheduler(); // This function never returns unless RTOS scheduler runs out of memory and
fails

  return 0;
}
```

Output -



Part 2:

Code:

This code contains extra credit also.

gpio_isr.h -

```
#pragma once
#include <stdio.h>

typedef enum {
  GPIO_INTERRUPT_FALLING_EDGE,
  GPIO_INTERRUPT_RISING_EDGE,
} gpio_interrupt_e;

typedef void (*function_pointer_t)(void);
void gpio0__attach_interrupt(uint32_t pin, gpio_interrupt_e interrupt, function_pointer_t callback);

void gpio0__interrupt_dispatcher(void);

int pin_check();

void pin_clr(int pin);
```

gpio_isr.c -

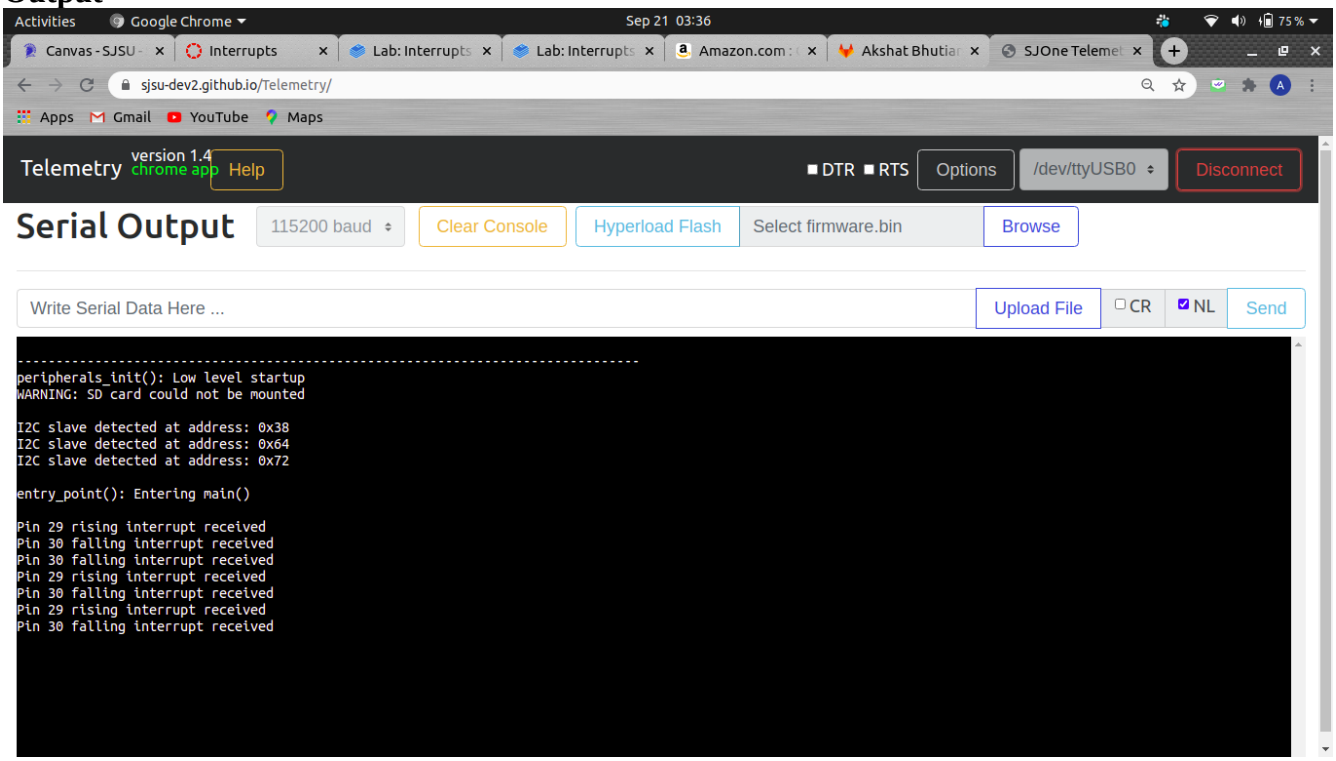
```
#include "gpio_isr.h"
#include "lpc40xx.h"

static function_pointer_t gpio0callbacks[32];

void gpio0__attach_interrupt(uint32_t pin, gpio_interrupt_e interrupt_type, function_pointer_t
callback) {

  gpio0callbacks[pin] = callback;
  // for Falling edge
  if (interrupt_type == GPIO_INTERRUPT_FALLING_EDGE) {
    LPC_GPIOINT->IO0IntEnF |= (1 << pin);
  }
  // for Rising edge
  if (interrupt_type == GPIO_INTERRUPT_RISING_EDGE) {
    LPC_GPIOINT->IO0IntEnR |= (1 << pin);
  }
}

// Function for clearing port 0 pin
void pin_clr(int num) { LPC_GPIOINT->IO0IntClr |= (1 << num); }
```

```
int pin_check() {
  for (int i = 0; i < 32; i++) {
    if ((LPC_GPIOINT->IO0IntStatF) & (1 << i)) {
      return i;
    }
    if ((LPC_GPIOINT->IO0IntStatR) & (1 << i)) {
      return i;
    }
  }
}

void gpio0__interrupt_dispatcher(void) {
  const int pin_num = pin_check();

  function_pointer_t attached_user_handle = gpio0callbacks[pin_num];

  attached_user_handle();

  pin_clr(pin_num);
}
```

main -

```
void pin29_isr(void) { fprintf(stderr, "\nPin 29 rising interrupt received"); }

void pin30_isr(void) { fprintf(stderr, "\nPin 30 falling interrupt received"); }
void main(void) {
  NVIC_EnableIRQ(GPIO_IRQn);

  LPC_GPIO0->DIR &= ~(1 << 29);
  LPC_GPIO0->DIR &= ~(1 << 30);

  gpio0__attach_interrupt(30, GPIO_INTERRUPT_RISING_EDGE, pin30_isr);
  gpio0__attach_interrupt(29, GPIO_INTERRUPT_FALLING_EDGE, pin29_isr);

  lpc_peripheral__enable_interrupt(LPC_PERIPHERAL__GPIO, gpio0__interrupt_dispatcher, NULL);
  vTaskStartScheduler();
}
```

Output -


Activities
Google Chrome
Sep 21 03:36
Canvas - SJSU -
Interrupts
Lab: Interrupts
Lab: Interrupts
Amazon.com :
Akshat Bhutia
SJOne Telemet
sjsu-dev2.github.io/Telemetry/
Apps
Gmail
YouTube
Maps
Telemetry
version 1.4
chrome app
Help
DTR
RTS
Options
/dev/ttyUSB0
Disconnect
Serial Output
115200 baud
Clear Console
Hyperload Flash
Select firmware.bin
Browse
Write Serial Data Here ...
Upload File
CR
NL
Send
---
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()
Pin 29 rising interrupt received
Pin 30 falling interrupt received
Pin 30 falling interrupt received
Pin 29 rising interrupt received
Pin 30 falling interrupt received
Pin 29 rising interrupt received
Pin 30 falling interrupt received