ME102b Final Report: Group 26

Cassie Young

Opportunity

Prevent the common bike/scooter thefts in Berkeley. Even bike U-locks are sometimes insufficient to prevent theft; currently existing solutions are mostly passive and do not actively deter thieves. Alarmed bike locks also exist but don't actively prevent the thief from taking the bike.

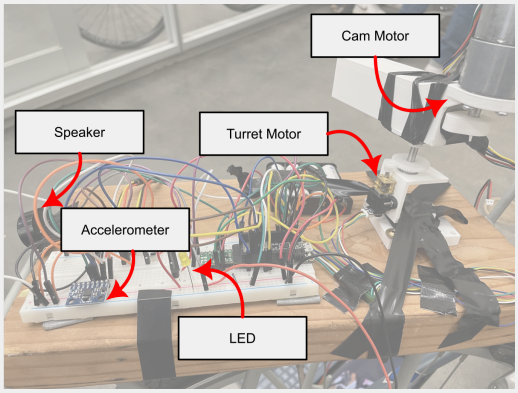
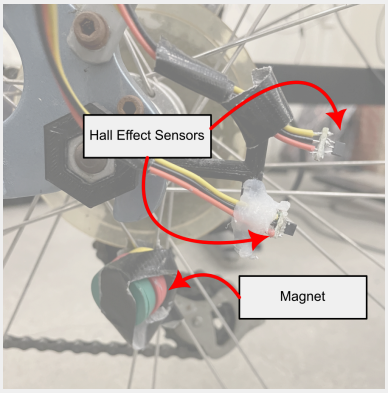
High-Level Strategy

We attached a Hall effect sensor and magnets to the rear wheel and also used an accelerometer to monitor motion, with a speaker and LED to sound an alarm if a theft is detected. We considered a theft attempt to be a detection of 2 complete revolutions of the rear wheel or more than 2 Gs of acceleration.

In addition to using alarms as theft deterrents, the system uses a method of offensive deterrent using pepper spray. Once a theft attempt is detected, a turret begins spraying pepper spray at its surroundings. We planned to have the turret have full pan/tilt functionality with two degrees of freedom covering all angles but due to time constraints only have pan functionality along one degree of freedom.

We initially planned for the security system to be armed via an IoT application on a smartphone, but ultimately did not implement due to trigger inconsistencies. Other intended but unimplemented IoT applications included the user and the police being notified through a smartphone app via a WiFi or cellular data connection. Finally, we also intended to implement a face tracking system attached to the turret using an ESP32-CAM, but the camera could not reliably identify a potential thief's face.

Integrated Physical Device

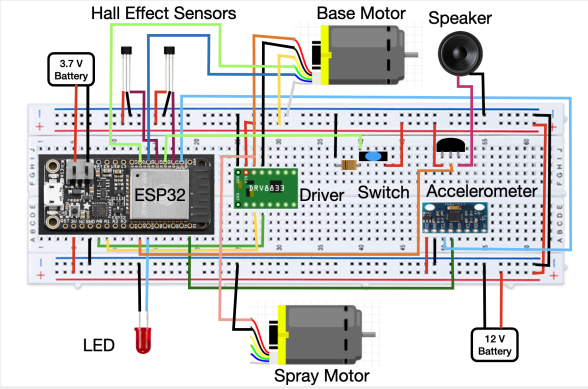
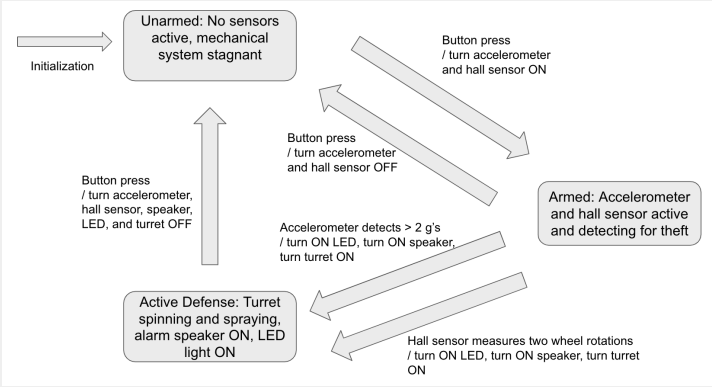
Figure 1	Figure 2
	
Integrated physical device. Select components labeled.	Hall effect sensors connecting to the main assembly. Note wheel-mounted magnets.

- **Actuator I** (rotation): The *Turret Motor*, in conjunction with the gear train, rotates the platform.
- **Actuator II** (linear transmission): The *Cam Motor* pushes on the nozzle horizontally to spray the deterrent onto the assailant.
- **Actuator III** (light and sound): The *LED* lights up and the *Speaker* emits a loud tone.
- **Sensor I** (rotational motion): *Hall Effect Sensors* detect the consecutive passing of wheel-mounted *Magnet*, then alerts the security system.
- **Sensor II** (linear motion): The *Accelerometer* detects sudden linear acceleration, which is passed to the security system.

Function-Critical Decisions & Calculations

Decision #1: turning the turret	Decision #2: actuating the sprayer
Measurements showed that the total mass of the components being rotated by the turret added up to 350g. Let: • m = total mass of the components being rotated • τ_m = stall torque provided by the Pololu kit micromotor • τ_r = torque required to turn the turret • α_r = desired angular acceleration • L_{xx} = moment of inertia of the components being rotated about the Z-axis (parallel to the output shaft) $m = 0.350$ kg $\alpha_r = 10\frac{rad}{s^2}$ $L_{xx} = 582668.42g \cdot mm^2 =$ $\tau = L \cdot \alpha$ $\tau_r = 582668.42g \cdot mm^2 \cdot 10\frac{rad}{s^2}$ $\tau_r = 0.00582$ N· m τ_m = 1.3 kg · cm = 0.013 kg · m, so even with a 1.5 factor of safety the micromotor provides enough torque to rotate the turret. A 1:3 gear ratio was still used in order to increase the torque to provide safety in case of frictional losses, losses in the power train and other inefficiencies.	Experimental data determined that the sprayer requires around 2kg of force applied to its head in order to depress the button and spray liquid. Let: • r = radius of cam • μ_f = coefficient of friction between the cam and the spray head • ϕ = pressure angle between cam and sprayer head • F_p = force needed to press the trigger and make it spray • tau_r = torque required of motor • tau_m = torque provided by motor $\mu_f = 0.42$ $\phi = 20$ r = 2.5 cm $F_p = 19.6$ N $= N$ $F_{frict} = \frac{\mu_f N}{cos(\phi)} = 20.172$ N $F_{total} = F_p + F_{frict} = 39.77$ N $\tau_r = F_{total} * r = 0.994N \cdot m$ $\tau_m = 1.765$ N · m, so the large kit motor is sufficiently strong to depress the button.

Circuit Diagram & State Transition Diagram

Figure 3	Figure 4
	
Circuit diagram	State Transition diagram

Reflection

To create a successful project, it is important for each member of the group to feel enthusiastic about the concept as that will foster creativity and eventually a final product that everyone is proud of. If we were to do something differently, we would try to complete tasks in a more timely manner. By having to rush with certain components of our design, it didn't turn out exactly as we had hoped. We would also assign specific, actionable tasks to people with strict deadlines to ensure people are held accountable for their tasks and finish them on time. We found that unfortunately, people do not complete tasks if not given specific goals in mind to achieve, so strictly planning who will accomplish what is a must-have.

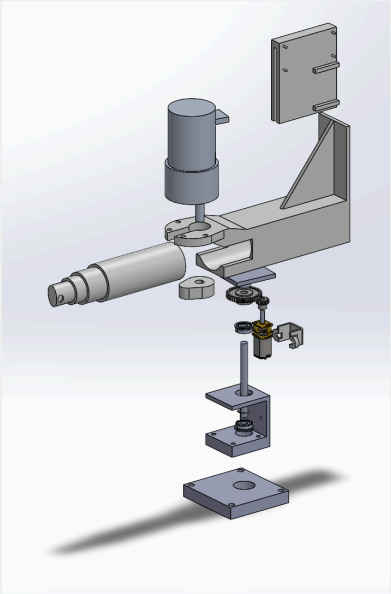
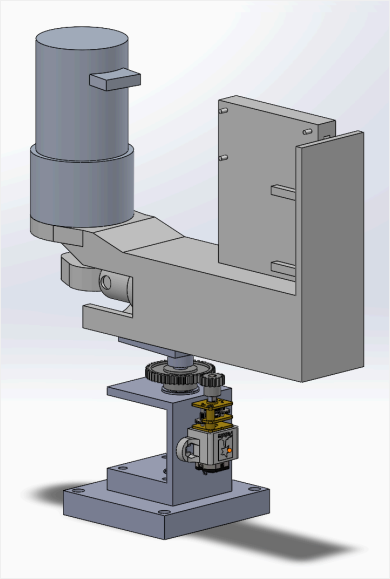
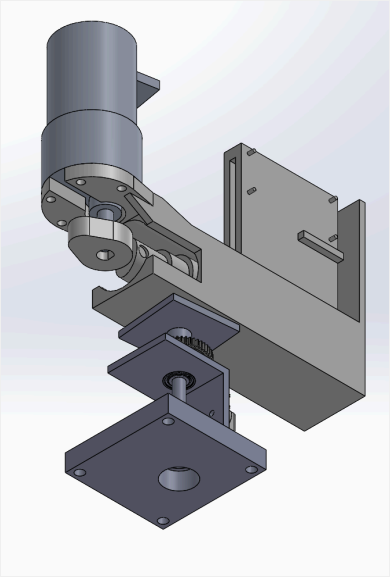
Appendices

Bill of Materials

Item Number	Name	Quantity	Unit Cost	Total Cost	Manufacturer	Supplier
5840	OV5640 Camera Breakout	1	$9.95	$9.95	Adafruit	Adafruit
3619	HUZZAH32 – ESP32 Feather Board	1	$21.95	$21.95	Adafruit	Adafruit
1898	PCB Mount Mini Speaker	1	$1.85	$1.85	Adafruit	Adafruit
3942	HC-SR04 Ultrasonic sensor	1	$3.95	$3.95	Adafruit	Adafruit
SKHLAAA010	Tactile Pushbutton	1	$0.00	$0.00	Alps Electric	Discontinued
634	Metal DC Geared Motor w/Encoder - 12V 251RPM	1	$29.00	$29.00	DFRobot	DFRobot
EP-A3144	Hall effect sensors	1	$6.99	$6.99	EPLZON	Amazon
generic	Spray Bottle	1	$0.00	$0.00	Fantasia IC	Household
B08TVLYB3Q	Zip ties	1	$5.99	$5.99	HAVE ME TD	Amazon
1953399	MPU9250 Accelerometer	1	$14.99	$14.99	HiLetgo	Amazon
	10mm M3 machine screws	4	$0.35	$1.40	Hillman	Ace Hardware
LY3330	LED Yellow Diffused 5mm	1	$0.15	$0.15	Jameco ValuePro	Jameco
DB-XZ-H1	Jumper wire kit	1	$9.99	$9.99	KABUDA	Amazon
21448	Retaining compound	1	$14.50	$14.50	Loctite	Amazon
SN12SF83QO H11452IC	GT-U7 GPS module	1	$10.99	$10.99	MakerFocus	Amazon

2810N1	Miter gear	3	$3.20	$9.60	McMaster	McMaster
2662N312	Gear	2	$4.26	$8.52	McMaster	McMaster
2662N31	Gear	2	$6.56	$13.12	McMaster	McMaster
98541A113	Retaining rings	1	$6.24	$6.24	McMaster	McMaster
generic	Magnets	4	$0.00	$0.00	N/A	Household
BK-3MCCA4BA	Eneloop Rechargable Batteries	4	$3.75	$15.00	Panasonic	Amazon
2215	Micro Metal Gearmotor	1	$20.95	$20.95	Pololu	Pololu
989	Micro Metal Gearmotor Bracket	1	$2.95	$2.95	Pololu	Pololu
2130	DRV8833 Dual Motor Driver	1	$9.95	$9.95	Pololu	Pololu
352	830-Point Breadboard	1	$5.95	$5.95	Pololu	Pololu
535218	Flanged bearing	2	$3.99	$7.98	ServoCity	ServoCity
2100-0005-0250	5mm shaft	1	$2.49	$2.49	ServoCity	ServoCity
10969	Resistor kit	1	$8.95	$8.95	SparkFun	Lab kits
PRT-13268	Protoboard	1	$8.95	$8.95	SparkFun	Sparkfun
ASR00007	Lithium Ion Polymer Battery	1	$5.95	$5.95	TinyCircuits	Digikey
22awg	22 AWG wire	1	$12.99	$12.99	TUOFENG	Amazon
			Grand Total	$271.29		

CAD Renders

Figure 5	Figure 6	Figure 7
		
Exploded view of the turret assembly	Rear view, showing the motor and gear train.	Ventral view, showing cam transmission

Code

```
// Basic demo for accelerometer readings from Adafruit MPU6050

// ESP32 Guide:
https://RandomNerdTutorials.com/esp32-mpu-6050-accelerometer-gyroscope-arduino/
// ESP8266 Guide:
https://RandomNerdTutorials.com/esp8266-nodemcu-mpu-6050-accelerometer-gyroscop
e-arduino/
// Arduino Guide:
https://RandomNerdTutorials.com/arduino-mpu-6050-accelerometer-gyroscope/

#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <ESP32Encoder.h>
#include <Wire.h>
#include <WiFi.h>
#include <TinyGPSPlus.h>
#define BTN 32
#define SPK 4
#define BIN_1 26
#define BIN_2 25
#define LED 13
```

```
#define LED_2 5
#define MOTOR_2 18

ESP32Encoder encoder;
ESP32Encoder encoder2;
// The TinyGPSPlus object
TinyGPSPlus gps;


//const char* ssid = "xtr";
//const char* password = "cracklepop!";

//WiFiServer server(80); // Port 80
Adafruit_MPU6050 mpu;

const int freq = 5000;
const int pwmChannel = 0;
const int resolution = 8;
int hallPin1 = 15;
int hallPin2 = 14;
int totalAccel = 0;
int wheel_revs = 0;
int event = 1;
// event 1 = disarmed, event 2 = armed, event 3 = alarming
int wait30 = 30000;
bool detected1 = false;
bool detected2 = false;
String buttonState = "";
volatile bool buttonIsPressed = false;

int omegaSpeed = 0;
int omegaDes = 7;
int omegaMax = 19;   // CHANGE THIS VALUE TO YOUR MEASURED MAXIMUM SPEED
int theta = 0;
int thetaDes = 0;
int thetaMax = 1500;
int D = 0;
int error = 0;
int dir = 1;

int Kp = 2;   // TUNE THESE VALUES TO CHANGE CONTROLLER PERFORMANCE
int Ki = 0.01;
int IMax = 0;
int SumError = 0;

//Setup interrupt variables ----------------------------
volatile int count = 0; // encoder count
volatile bool interruptCounter = false;    // check timer interrupt 1
volatile bool deltaT = false;     // check timer interrupt 2
```

```
int totalInterrupts = 0;   // counts the number of triggering of the alarm
hw_timer_t * timer0 = NULL;
hw_timer_t * timer1 = NULL;
portMUX_TYPE timerMux0 = portMUX_INITIALIZER_UNLOCKED;
portMUX_TYPE timerMux1 = portMUX_INITIALIZER_UNLOCKED;

// setting PWM properties ----------------------------
const int ledChannel_1 = 1;
const int ledChannel_2 = 2;
const int MAX_PWM_VOLTAGE = 255;
const int NOM_PWM_VOLTAGE = 150;


void hall_detect_1()
{
  detected1 = true;
}

void hall_detect_2()
{
  detected2 = true;
}


void wheel_turn()
{
  if (detected1 && detected2){
    detected1 = false;
    detected2 = false;
    wheel_revs += 1;
    Serial.println(wheel_revs);
  }
}


void IRAM_ATTR isr() {  // the function to be called when interrupt is
triggered
  buttonIsPressed = true;
}

//Initialization -----------------------------------
void IRAM_ATTR onTime0() {
  portENTER_CRITICAL_ISR(&timerMux0);
  interruptCounter = true; // the function to be called when timer interrupt is
triggered
  portEXIT_CRITICAL_ISR(&timerMux0);
}

void IRAM_ATTR onTime1() {
```

```
  portENTER_CRITICAL_ISR(&timerMux1);
  count = encoder.getCount( );
  encoder.clearCount ( );
  deltaT = true; // the function to be called when timer interrupt is triggered
  portEXIT_CRITICAL_ISR(&timerMux1);
}

void setup() {
  Serial.begin(115200);
  Serial2.begin(9600);
  pinMode(LED, OUTPUT);
  pinMode(LED_2, OUTPUT);
  ledcSetup(pwmChannel, freq, resolution);
  ledcAttachPin(SPK, pwmChannel);

  ////////////////////////////////////
  ESP32Encoder::useInternalWeakPullResistors = UP; // Enable the weak pull up
resistors
  encoder.attachHalfQuad(33, 27); // Attache pins for use as encoder pins
  encoder.setCount(0);  // set starting count value after attaching

  // configure LED PWM functionalitites
  ledcSetup(ledChannel_1, freq, resolution);
  ledcSetup(ledChannel_2, freq, resolution);

  // attach the channel to the GPIO to be controlled
  ledcAttachPin(BIN_1, ledChannel_1);
  ledcAttachPin(BIN_2, ledChannel_2);

    // initilize timer
  timer0 = timerBegin(0, 80, true);  // timer 0, MWDT clock period = 12.5 ns *
TIMGn_Tx_WDT_CLK_PRESCALE -> 12.5 ns * 80 -> 1000 ns = 1 us, countUp
  timerAttachInterrupt(timer0, &onTime0, true); // edge (not level) triggered
  timerAlarmWrite(timer0, 1500000, true); // 1500000 * 1 us = 1.5 s, autoreload
true

  timer1 = timerBegin(1, 80, true);  // timer 1, MWDT clock period = 12.5 ns *
TIMGn_Tx_WDT_CLK_PRESCALE -> 12.5 ns * 80 -> 1000 ns = 1 us, countUp
  timerAttachInterrupt(timer1, &onTime1, true); // edge (not level) triggered
  timerAlarmWrite(timer1, 10000, true); // 10000 * 1 us = 10 ms, autoreload
true

  // at least enable the timer alarms
  timerAlarmEnable(timer0); // enable
  timerAlarmEnable(timer1); // enable
//  // Connect to wifi.
//  Serial.println();
//  Serial.print("Connecting with ");
//  Serial.println(ssid);
```

```
//
//  WiFi.begin(ssid, password);
//
//  while (WiFi.status() != WL_CONNECTED) {
//    delay(500);
//    Serial.print(".");
//  }
//  Serial.println("Connected with WiFi.");
//
//  // Start Web Server.
//  server.begin();
//  Serial.println("Web Server started.");
//
//  // print ip
//  Serial.print("http://");
//  Serial.println(WiFi.localIP());

  attachInterrupt(BTN, isr, RISING);
  attachInterrupt(hallPin1, hall_detect_1, RISING);
  attachInterrupt(hallPin2, hall_detect_2, RISING);

  while (!Serial)
    delay(10); // will pause Zero, Leonardo, etc until serial console opens

  Serial.println("Adafruit MPU6050 test!");

  // Try to initialize!
  if (!mpu.begin()) {
    Serial.println("Failed to find MPU6050 chip");
    while (1) {
      delay(10);
    }
  }
  Serial.println("MPU6050 Found!");

  mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
  mpu.setGyroRange(MPU6050_RANGE_500_DEG);
  mpu.setFilterBandwidth(MPU6050_BAND_5_HZ);
}


void loop() {
  /* Get new sensor events with the readings */
  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);

  totalAccel = pow((pow(a.acceleration.x, 2) + pow(a.acceleration.y, 2) +
pow(a.acceleration.z, 2)), 0.5);
```

```
//  if (wheel_turn()){
//    Serial.println("wheel turned");
//  }
//  if (totalAccel > 20){
//    Serial.println(totalAccel);
//  }

////////////////////////////////////////////////////////
// README!!!!!
// Commenting out the wireless arming system code for now.
// Seems to only (for some reason) run the state machine
// code when a button on the website is pressed, i.e. it
// won't detect when accel/encoders are tripped when
// button is not clicked (!!). Will try working on it
// after A5 is turned in.
////////////////////////////////////////////////////////


//  // If disconnected, try to reconnect every 30 seconds.
//  if ((WiFi.status() != WL_CONNECTED) && (millis() > wait30)) {
//    Serial.println("Trying to reconnect WiFi...");
//    WiFi.disconnect();
//    WiFi.begin(ssid, password);
//    wait30 = millis() + 30000;
//  }
//  // Check if a client has connected..
//  WiFiClient client = server.available();
//  if (!client) {
//    return;
//  }
//
////  Serial.print("New client: ");
////  Serial.println(client.remoteIP());
//
//  // Espera hasta que el cliente envíe datos.
//  // while(!client.available()){ delay(1); }
//
//  ////////////////////////////////////////////////////
//  // Read the information sent by the client.
//  String req = client.readStringUntil('\r');
//  Serial.println(req);
//
//  // Make the client's request.
//  if (req.indexOf("check") != -1){
//    if (event == 2){
//      buttonState = "System is ARMED";
//    } else if (event == 3) {
//      buttonState = "System is ALARMING";
//    } else {
```

```
//      buttonState = "System is DISARMED";
//    }
//  }
//
//  ////////////////////////////////////////////
//  //  WEB PAGE. ///////////////////////////
//  client.println("HTTP/1.1 200 OK");
//  client.println("Content-Type: text/html");
//  client.println(""); //  Important.
//  client.println("<!DOCTYPE HTML>");
//  client.println("<html>");
//  client.println("<head><meta charset=utf-8></head>");
//  client.println("<body><center><font face='Arial'>");
//  client.println("<h1>Servidor web con ESP32.</h1>");
//  client.println("<h2><font color='#009900'>Credit to example code: KIO4.COM
- Juan A. Villalpando</font></h2>");
//  client.println("<br><br>");
//  client.println("<a href='on2'><button>Click to ARM security
system</button></a>");
//  client.println("<a href='off2'><button>Click to DISARM security
system</button></a>");
//  client.println("<a href='check'><button>Check armed state</button></a>");
//  client.println("<br><br>");
//  client.println(buttonState);
//  client.println("</font></center></body></html>");
//
////  Serial.print("Client disconnected: ");
////  Serial.println(client.remoteIP());
//  client.flush();
//  client.stop();

//Serial.println(wheel_revs);

  ////////////////////////////////////
  //motor stuff
  // READ ME
  // for some reason the direction change
  // stuff doesnt work when this code is in
  // the switch statement

  if (event == 3){
    if (interruptCounter) {
    portENTER_CRITICAL(&timerMux0);
    interruptCounter = false;
    portEXIT_CRITICAL(&timerMux0);

    totalInterrupts++;

    if (totalInterrupts%2 == 0) {
```

```
      thetaDes = 1;
    }
    if (totalInterrupts%2 == 1){
      thetaDes = thetaMax;
    }
  }

  if (deltaT) {
      portENTER_CRITICAL(&timerMux1);
      deltaT = false;
      portEXIT_CRITICAL(&timerMux1);

      theta += count;
//      potReading = analogRead(POT);


      //A6 CONTROL SECTION
      //CHANGE THIS SECTION FOR P AND PI CONTROL

      error = thetaDes - theta;
      int P = Kp * error;
      SumError += error;
      if (SumError > 100){
        SumError = 100;
      } else if (SumError < -100){
        SumError = -100;
      }

      D = Kp * (error + (SumError / 25));

      //END A6 CONTROL SECTION

      //Ensure that you don't go past the maximum possible command
      if (D > MAX_PWM_VOLTAGE) {
          D = MAX_PWM_VOLTAGE;
      }
      else if (D < -MAX_PWM_VOLTAGE) {
          D = -MAX_PWM_VOLTAGE;
      }

      //Map the D value to motor directionality
      //FLIP ENCODER PINS SO SPEED AND D HAVE SAME SIGN
      if (D > 0) {
          ledcWrite(ledChannel_1, LOW);
          ledcWrite(ledChannel_2, D);
      }
      else if (D < 0) {
          ledcWrite(ledChannel_1, -D);
          ledcWrite(ledChannel_2, LOW);
```

```
      }
      else {
          ledcWrite(ledChannel_1, LOW);
          ledcWrite(ledChannel_2, LOW);
      }

      plotControlData();
  }
  }
  ////////////////////////////////

//  if (event == 3){
//
//  }


  wheel_turn();

  switch (event){
    case 1:
//      if (req.indexOf("on2") != -1) { // when you press the button on the
webpage
      if (CheckForButtonPress()){
        Serial.println("Button pressed - system armed\n");
        detected1 = false;
        detected2 = false;
//        Serial.println(event);
//        Serial.println("");
        digitalWrite(LED, HIGH);
        event = 2;

      }
      break;
    case 2:
//      Serial.println(totalAccel);

//      if (hall_detect_1){
//        Serial.println("hall 1 detect");
//      }
//
//      if (hall_detect_2){
//        Serial.println("hall 2 detect");
//      }

      if (totalAccel > 15 || wheel_revs > 3){
        // more than 2 g's total sensed by accelerometer
        // note: sensor by default includes force of gravity (aka 10 = 10
m/s^2)
        digitalWrite(LED, LOW);
```

```
        led_alarm_on();
        speaker_alarm_on();
        gps_location_start();
//        spray_motor_on();
        wheel_revs = 0;
        spray_on();
        Serial.println("Spray on");
//        delay(1000); // remove later
        event = 3;
//        Serial.println(event);
//        Serial.println("");
      }

//      if (req.indexOf("off2") != -1){ // when off button pressed
      if (CheckForButtonPress()){
        Serial.println("Button pressed - system disarmed");
        digitalWrite(LED, LOW);
        led_alarm_off();
        speaker_alarm_off();
        spray_motor_off();
        gps_location_stop();
        spray_off();
        Serial.println("Spray off");
        Serial.println("");
        wheel_revs = 0;
        event = 1;
      }
      break;
    case 3:
//      if (req.indexOf("off2") != -1){ // off button pressed
      if (CheckForButtonPress()){
        Serial.println("Button pressed - system disarmed");
        digitalWrite(LED, LOW);
        led_alarm_off();
        speaker_alarm_off();
        gps_location_stop();
        spray_motor_off();
        Serial.println("Spray off");
        spray_off();
        wheel_revs = 0;
        Serial.println("");
        event = 1;
      }
      break;
  }
//  if (wheel_turn()){
//    Serial.println("turn");
//  }
//  Serial.println(totalInterrupts);
```

```
}



void led_alarm_on()
{
  digitalWrite(LED_2, HIGH);
}
//void speaker_alarm_on()
//{
//  Serial.println("Speaker alarming");
//}
void gps_location_start()
{
  Serial.println("GPS begins transmitting location data");
}
void led_alarm_off()
{
  digitalWrite(LED_2, LOW);
}
//void speaker_alarm_off()
//{
//  Serial.println("Speaker stops alarming");
//}
void gps_location_stop()
{
  Serial.println("GPS stops transmitting location data");
}

void pan_start()
{
  Serial.println("State 2->3: Begin spinning the rotating motor");
}
void pan_stop()
{
  Serial.println("State 3->2 or 3->1: Stop spinning the rotating motor");
}
void speaker_alarm_on() {
  ledcAttachPin(SPK, pwmChannel);
  ledcWriteTone(pwmChannel, 600);
}

void speaker_alarm_off() {
  ledcDetachPin(SPK);
  ledcWriteTone(pwmChannel, 0);
}
bool CheckForButtonPress() {
  if (buttonIsPressed == true){
```

```
    buttonIsPressed = false;
    return true;
  }
  else {
    return false;
  }
}

void displayInfo()
{

  Serial.print(F("Location: "));

  if (gps.location.isValid()){
    Serial.print("Lat: ");
    Serial.print(gps.location.lat(), 6);
    Serial.print(F(","));
    Serial.print("Lng: ");
    Serial.print(gps.location.lng(), 6);
    Serial.println();
  }
  else
  {
    Serial.print(F("INVALID"));
  }
}

void updateSerial()
{
  delay(500);
  while (Serial.available())
  {
    Serial2.write(Serial.read());//Forward what Serial received to Software
Serial Port
  }
  while (Serial2.available())
  {
    Serial.write(Serial2.read());//Forward what Software Serial received to
Serial Port
  }
}
void plotControlData() {
  Serial.print("Position:");
  Serial.print(theta);
  Serial.print(" ");
  Serial.print("Desired_Position:");
  Serial.print(thetaDes);
  Serial.print(" ");
  Serial.print("PWM_Duty:");
```

```
  Serial.println(D);
  Serial.print(" ");
}

//void spray_motor_on(){
//    if (interruptCounter) {
//      portENTER_CRITICAL(&timerMux0);
//      interruptCounter = false;
//      portEXIT_CRITICAL(&timerMux0);
//
//      totalInterrupts++;
//
//      if (totalInterrupts%2 == 0) {
//        dir = -1*dir;
//      }
//    }
//    if (deltaT) {
//      portENTER_CRITICAL(&timerMux1);
//      deltaT = false;
//      portEXIT_CRITICAL(&timerMux1);
//
//      omegaSpeed = count;
////      potReading = analogRead(POT);
//      omegaDes = dir*omegaMax;
//
//      //A6 CONTROL SECTION
//      int error = omegaDes - omegaSpeed;
//      int P = Kp * error;
//      SumError += error;
//      if (SumError > 500){
//        SumError = 500;
//      } else if (SumError < -500){
//        SumError = -500;
//      }
//
//      D = Kp * (error + (SumError / 100));
//      //CHANGE THIS SECTION FOR P AND PI CONTROL
//
////      D = map(omegaDes, -omegaMax, omegaMax, -NOM_PWM_VOLTAGE,
NOM_PWM_VOLTAGE);  // REPLACE THIS LINE WITH P/PI CONTROLLER CODE
//
//      //END A6 CONTROL SECTION
//
//      //Ensure that you don't go past the maximum possible command
//      if (D > MAX_PWM_VOLTAGE) {
//          D = MAX_PWM_VOLTAGE;
//      }
//      else if (D < -MAX_PWM_VOLTAGE) {
//          D = -MAX_PWM_VOLTAGE;
```

```
//      }
//
//      //Map the D value to motor directionality
//      //FLIP ENCODER PINS SO SPEED AND D HAVE SAME SIGN
//      if (D > 0) {
//          ledcWrite(ledChannel_1, LOW);
//          ledcWrite(ledChannel_2, D);
//      }
//      else if (D < 0) {
//          ledcWrite(ledChannel_1, -D);
//          ledcWrite(ledChannel_2, LOW);
//      }
//      else {
//          ledcWrite(ledChannel_1, LOW);
//          ledcWrite(ledChannel_2, LOW);
//      }
//      plotControlData();
//  }
//}

void spray_motor_off(){
    ledcWrite(ledChannel_1, LOW);
    ledcWrite(ledChannel_2, LOW);
}
void spray_on(){
  digitalWrite(MOTOR_2, HIGH);
}
void spray_off(){
  digitalWrite(MOTOR_2, LOW);
}
```