Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Section: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Names: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ and \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| LABORATORY CHECK OFFS | |
| **Section 1 Calibrating Motor Offsets**  Calibration routine succeeds, robot lurches twice, once for each motor. PWM offset values recorded.  PWM offset for AB motor \_\_\_\_\_\_\_\_\_\_\_\_\_\_  PWM offset for CD motor \_\_\_\_\_\_\_\_\_\_\_\_\_\_  When battery is switched off, calibration routine fails, LED flashes RED/BLUE and serial monitor error message is printed.  **Section 2 Going Forward Using PWM Offsets and Gyro Sensor Feedback**  Robot goes forward in a straight line and corrects its heading if bumped.  **Section 3 Going Forward, Right, Backward, Left**  Robot goes forward, right turn, backward, left turn. | \_\_\_\_  \_\_\_\_  \_\_\_\_  \_\_\_\_ |
| Points | \_\_\_\_ |

Prelab:

No prelab work required.

Learning Outcomes:

By the end of this lab you will be able to:

1. Control the robot’s direction using the gyroscope’s yaw angle as feedback

Overview:

Making the robot go forward in a straight line or turn exactly 90 degrees can be very difficult using only pre-programmed pwm values and time durations. It is much easier to control the robot if a feedback signal is available to tell the robot what direction it is actually heading. In this lab, you will use the gryroscope yaw angle to tell the robot exactly what direction it is heading and to help it correct any errors. You will be closing a feedback control loop that will change the PWM values to each motor independently so that just the right amount of voltage is applied to each motor to make it move forward.

**SECTION 1 – Calibrating Motor Offsets - Determining the minimum PWM value / motor voltage to make each motor turn.**

Applying different pwm values to the motor is essentially the same as applying different dc voltages to the motor. The motor has friction which prevents it from turning until enough dc voltage is applied to the motor. This value of friction when the motor is called stiction (think of motor being stuck by friction i.e. stiction).

In this section you will determine the minimum PWM value to turn each motor, noting that the value can be very different for each motor. The minimum pwm value will be stored as an offset for each motor’s commanded pwm value.

Recall that a pwm value range from 0 to 255 causes a dc voltage value from 0 to 9v (the total battery pack voltage) to be applied to the motor. The dc voltage range is only approximate due to drops in the motor driver IC (lm298).

**Procedure:**

1) Start with you code from last week’s lab, including the gyro sensor code. Your code should be printing out the yaw value.

2) Add a new state to your list of enumerated states and call it CALIBRATE\_MOTOR\_STICTION. Make this state the last state in the list of enumerated states

3) Set your state machine so that it wakes up in this state.

directionState\_t directionState=CALIBRATE\_MOTOR\_STICTION;

4) At the top of your code add the following variable declarations. Note that the names used correspond directly to the labels on the motor driver board. If you have followed the wiring color code in the previous labs, everything should work. AB refers to one motor, CD refers to the other motor. The calibration routine for stiction will run only once on startup of the robot and the code does not have to run fast. Therefore, it is acceptable to write the code using digitalWrite() type commands and delays so that the code is more readable.

//motor driver pins

const int AB\_mtr\_INA\_PIN = 10;

const int AB\_mtr\_INB\_PIN = 9;

const int CD\_mtr\_INC\_PIN = 8;

const int CD\_mtr\_IND\_PIN = 7;

const int PWM\_AB\_PIN = 6;

const int PWM\_CD\_PIN = 5;

int MotorABpwmOffset = 0;

int MotorCDpwmOffset = 0;

5) In the switch-case statement, add the following state code.

case CALIBRATE\_MOTOR\_STICTION:

//entry housekeeping

if (isNewState) {

//find\_motorstiction\_using\_Gyro();

MotorABpwmOffset = find\_motorstiction\_using\_gyro(AB\_mtr\_INA\_PIN, AB\_mtr\_INB\_PIN, PWM\_AB\_PIN);

Serial.print("Motor Phase offset is "); Serial.println(MotorABpwmOffset);

display\_color\_on\_RGB\_led(DIM\_GREEN\_COLOR);

delay(500);

MotorCDpwmOffset = find\_motorstiction\_using\_gyro(CD\_mtr\_INC\_PIN, CD\_mtr\_IND\_PIN, PWM\_CD\_PIN);

Serial.print("Motor Phase offset is "); Serial.println(MotorCDpwmOffset);

display\_color\_on\_RGB\_led(MED\_BLUE\_COLOR);

delay(500);

}

//state business

//exit housekeeping

directionState=STOPPED;

break;

6) Next, you will define a new procedure called find\_motorstiction\_using\_gyro(). Put the procedure definition at the bottom of your code with the other procedures such as display\_color\_on\_RGB\_led().

This new procedure will need to have input arguments (look at how the procedure is called in the sample code above). Think through the logic of what the procedure has to do and figure out how the input arguments map to the function of the procedure, then add any necessary code. You will need to read through the rest of this section first to understand the logic.

In the procedure, the pwm value applied to one motor is ramped up slowly until eventually the motor turns. This will cause the robot to move. Use a delay for 20 milliseconds in the while loop to slow down the rate at which new PWM values are tried.

while (calibrating\_motor\_stiction) {

pwm\_value\_sent\_to\_motor++;

delay(20); //was 5

The robot’s motion can be detected by observing a sudden change in the yaw rate value from the gryo sensor. A rotational speed of 2 degrees per second in yaw in either direction (positive or negative) indicates that the pwm value is enough to overcome stiction.

if (abs(normalizedGyroDPS.ZAxis) > 2) calibrating\_motor\_stiction = false;

If calibration succeeds, calibration stops and the procedure find\_motorstiction\_using\_gyro() will return an integer value which is the pwm offset value required to make the motor turn.

return pwm\_value\_sent\_to\_motor;

If calibration fails, a debug message is printed to the serial port and the led flashes red and blue and the motor pwm is set to zero. The microcontroller will need to be reset to attempt calibration again.

if (pwm\_value\_sent\_to\_motor > 250) //failed calibration

{ Serial.print(F("Calibration failed. Check if battery is connected/switched on."));

analogWrite(EnPWMPin, 0); // send pwm value to motor

while (1) {

display\_color\_on\_RGB\_led(DIM\_RED\_COLOR);

delay(100);

display\_color\_on\_RGB\_led(DIM\_BLUE\_COLOR);

delay(100);

}

}

By calling the procedure twice in the state machine state CALIBRATE\_MOTOR\_STICTION, with different values for the pin numbers, the offset for each motor can be found.

MotorABpwmOffset = find\_motorstiction\_using\_gyro(AB\_mtr\_INA\_PIN, AB\_mtr\_INB\_PIN, PWM\_AB\_PIN);

MotorCDpwmOffset = find\_motorstiction\_using\_gyro(CD\_mtr\_INC\_PIN, CD\_mtr\_IND\_PIN, PWM\_CD\_PIN);

These offsets are later used in the pwm commands to the motors. For now, just get the pwm offset values to print out and calibration to succeed.

7) Upload the code and make any changes required to fix bugs e.g. define missing constants, fix variables names, include other helper functions. Expect that there may be bugs. Use your troubleshooting skills to get the code working.

8) Print out the calibration values for each motor and record the values on the signoff sheet. Demonstration the calibration of the robot motors to your instructor.

**SECTION 2 – Going Forward Using PWM Offsets and Gyro Sensor Feedback**

In this section, the robot will be controlled to move forward in a straight line using feedback from the gyro sensor yaw angle and the pwm offsets for motors AB and CD calculated in the last section.

**Procedure:**

1. Declare a global variable for the direction the robot should be heading.

int robot\_commanded\_heading = 0;

1. Modify the FORWARD state as shown in the code box below so that the robot commanded heading equals the current yaw angle. The robot will continue to move in this direction for the entire state.

case FORWARD:

//entry housekeeping

if (isNewState) {

robot\_commanded\_heading = yaw;

Serial.print("new state is FORWARD, \tcommanded heading is ");

Serial.println(robot\_commanded\_heading);

display\_color\_on\_RGB\_led(DIM\_GREEN\_COLOR);

}

//state business

robot\_goes\_forward\_at\_given\_yaw\_at\_speed(robot\_commanded\_heading, 15);

//exit housekeeping

break;

1. Create a new procedure called robot\_goes\_forward\_at\_given\_yaw\_at\_speed(robot\_commanded\_heading, rate) with the following code. Rate is the speed at which the robot moves forward.

//================================================================

void robot\_goes\_forward\_at\_given\_yaw\_at\_speed(float yaw\_heading, int rate) {

go\_forward(rate, 3.0 \* (yaw\_heading - yaw)); // rate, steering amount

}

1. Modify your code for go\_forward() as shown below.

//==============================================================

void go\_forward(int rate, int steering) {

digitalWrite(AB\_mtr\_INA\_PIN, HIGH);

digitalWrite(AB\_mtr\_INB\_PIN, LOW);

digitalWrite(CD\_mtr\_INC\_PIN, HIGH);

digitalWrite(CD\_mtr\_IND\_PIN, LOW);

analogWrite(PWM\_AB\_PIN, constrain(MotorABpwmOffset + rate - steering, 0, 255));

analogWrite(PWM\_CD\_PIN, constrain(MotorCDpwmOffset + rate + steering, 0, 255));

}

1. You may have to change the direction pin levels HIGH/LOW to get your robot to move forward and not just turn in circles. This will depend on how your motors are wired (which lead is red or black at the motor and at the motor driver board). Make adjustments to the LOW or HIGH levels as needed based on your testing.
2. Put all the code pieces together so that your robot first calibrates the motor offsets (MotorABpwmOffset) and when it finishes the state CALIBRATE\_MOTOR\_STICTION, it moves on to going forward in straight line. E.g. change the exit housekeeping for the CALIBRATE\_MOTOR\_STICTION state so that the new directionState is FORWARD.
3. This may require debugging. Remember to use the serial monitor to print variables and check each step of the process to ensure it is working as expected.
4. You may have to make changes to the code to get this working.
5. Demonstrate that your robot moves forward in a straight line and if bumped, will correct itself back to the original heading/direction.
6. Get a sign off from your lab instructor.

**SECTION 3 – Turning Using PWM Offsets and Gyro Sensor Feedback**

Using gyroscope sensor feedback enables the robot platform to make precise 90 degree turns. In this section, you will build code to turn the robot left of right.

**Procedure:**

1. Define right turn and left turn states so that the robot turns 90 degrees. Set the commanded heading to yaw-90 or yaw+90 as appropriate.
2. In the right turn state, set the RGB led color to BRIGHT GREEN COLOR and define a constant for this color.
3. In the left turn state, set the RGB led color to BRIGHT RED COLOR and define a constant for this color.
4. Test the code by having the robot move through this sequence: FORWARD, RIGHT TURN, BACKWARD, LEFT TURN

case RIGHT\_TURN:

//entry housekeeping

if (isNewState) {

robot\_commanded\_heading = yaw - 90;

Serial.print(F("new state is RIGHT\_TURN,\tcommanded heading is "));

Serial.println(robot\_commanded\_heading);

display\_color\_on\_RGB\_led(DIM\_RED\_GREEN\_COLOR);

}

//state business

robot\_turns\_to\_heading(robot\_commanded\_heading);

//exit housekeeping

// if 2 seconds have passed and yaw=90, change directionState to BACKWARD

break;