Senior Design
ENG EC 463

To: Professor Pisano, Professor Alshaykh, Professor Hirsch

From: Christopher Liao, Anton Paquin, Jeffrey Lin, Eduardo Portet, Aviva Englander

Team: 15 - Laser Trac

Date: 11/19/17

Subject: First Deliverable Test Report

---

1. Project Objective

1.1. The overall objective of our project is to create a optical based tracking and communication system from a ground station to a UAV, allowing users to interact with their vehicles without using Radio Frequency communication.

2. Test Objective and Significance

2.1. Our first two tests were significant because they demonstrated our ability to complete two fundamental areas of our project. First and foremost we demonstrated our ability to use optical wireless communication. The objective of this test was to prove that we could send bits using a laser and receive and read them using receiver. This is important because until we have working optical wireless communication, tracking the drone with a laser is a useless exercise. Once we achieve this first objective, we can focus on the more difficult problem at hand, our tracking mechanism.

2.2. Our second test focused on achieving a rough example of a tracking mechanism. With this test we demonstrated an ability to track an object with a laser while utilizing a photodiode array in the way we hope to for our final project. We started out with tracking an object moving left to right, which only required shifting the azimuthal angle of the rotating laser. This is a significant first step in the tracking problem. The most important part of this test is how we incorporated the photodiode array into the tracking process. This is a technique we hope to use in our final solution to the laser tracking problem.

3. Equipment and Setup

3.1. Receiver Circuit

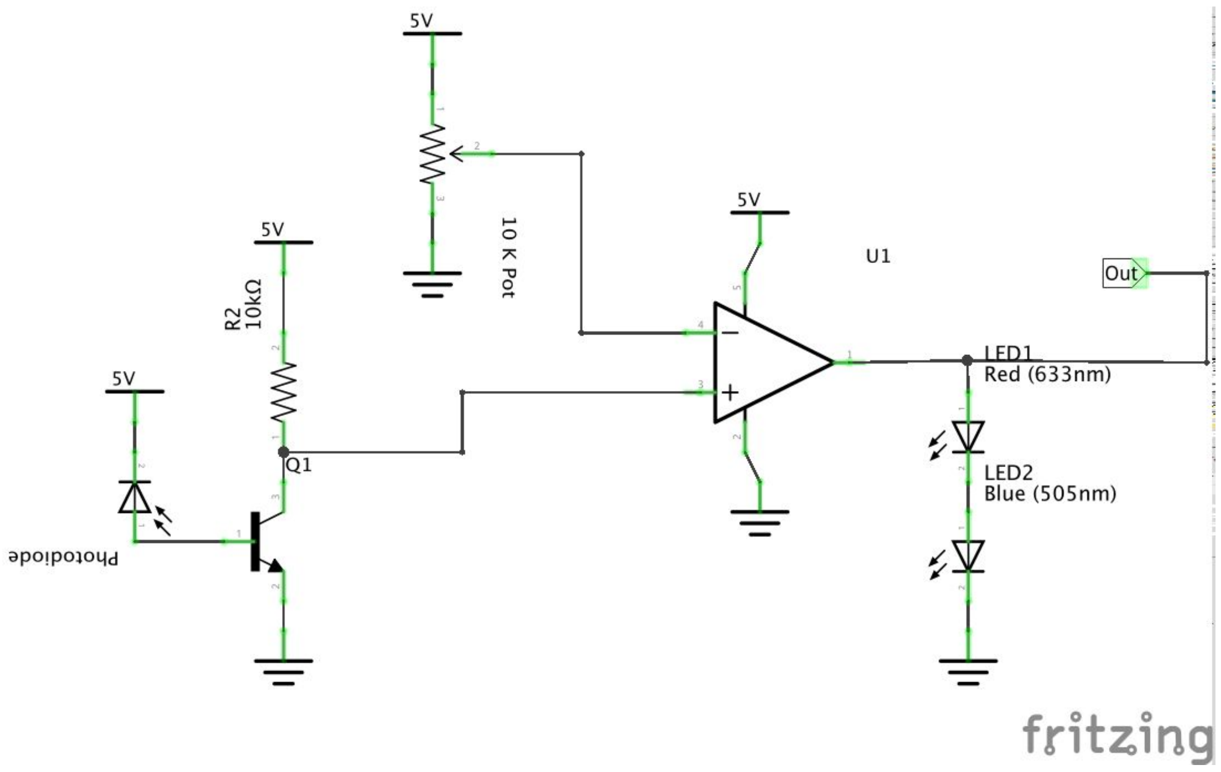


Figure 1. This is the circuit diagram of the photodiode circuit. The output is a digital value. The leds on the right act as indicators. When the photodiode sees light, the output is HIGH, otherwise it is LOW. 5V power provided by wall adapter.

3.2. Laser Control Circuit

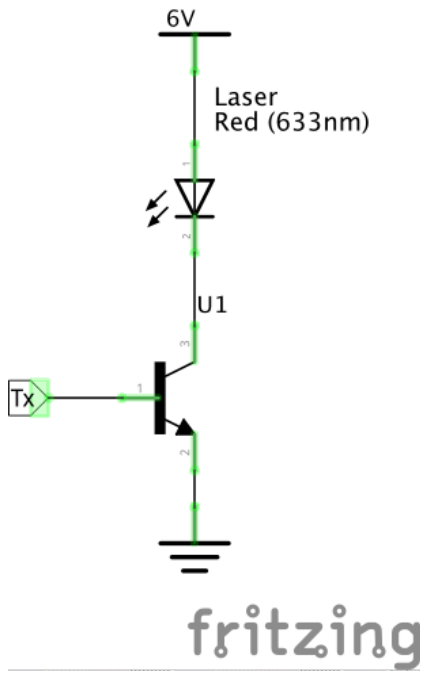


Figure 2. The laser is switched on and off by the Tx port using an NPN transistor. This circuit can be directly connected to the Tx wire of a serial port. 6V provided by 4 AAA batteries.

3.3. Stepper Motor Circuit

The stepper motor is not what we will use in the final product, but it is used for this preliminary test. The stepper motor we use is a cheap unipolar motor 28BYJ. The motor has 64 steps per revolution and is geared 64:1. This means that in half-step mode (which we use), there are 4096 steps per revolution.

The ideal controller program would use timer interrupts to “step” the motor, so that the Arduino is free to do other calculations while driving the motor. However, for this test, we use delay instructions in the code to delay the appropriate amount between steps. The delay amount we set for this test is 5 milliseconds. This means that during the tracking test, the motor rotates at approximately 2.9 rpm. To change the direction of a stepper motor, simply go through the step sequence in reverse.

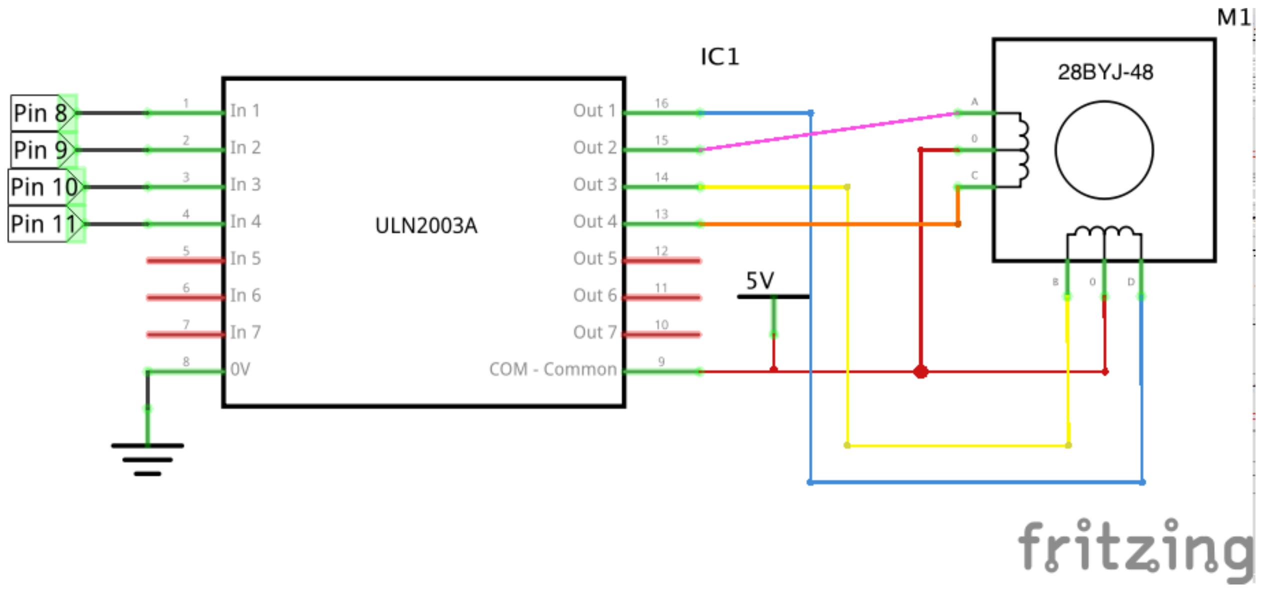


Figure 3. This diagram shows control of a unipolar stepper motor (a 28BYJ-48 in our case) with a transistor array (ULN2003A). 5V power provided by Arduino, but would ideally be powered separately.

3.4. Block Diagram of Echo test (communication)

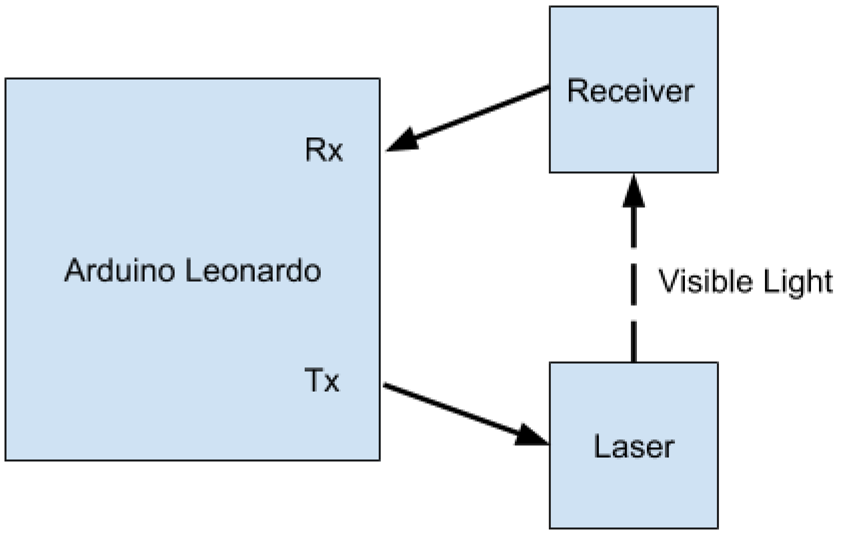


Figure 4. This shows how the echo back test is set up. A message sent on Tx is transmitted using the laser circuitry. The laser is aimed directly at the photodiode in the receiver circuit. Rx is then connected to the out port in the receiver circuit.

The laser and the photodiode receiver are positioned 5 centimeters apart.

3.5. Block Diagram of Tracking test

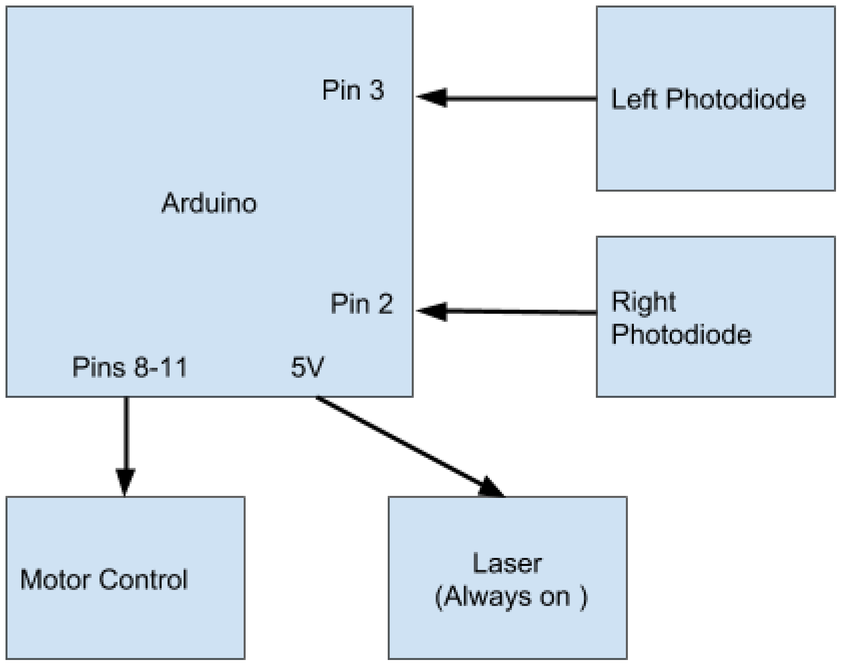


Figure 5. This shows how the Tracking test is setup. The Arduino controls the motor and receives input from the photodiodes. Pin 3 and 2 are set up as interrupt pins which trigger on the rising edge (when the photodiode sees light). The laser is always on. When the laser hits the left photodiode, the Arduino tells the motor to move right, and vice versa.

4. Measurements

4.1. Echo back Test

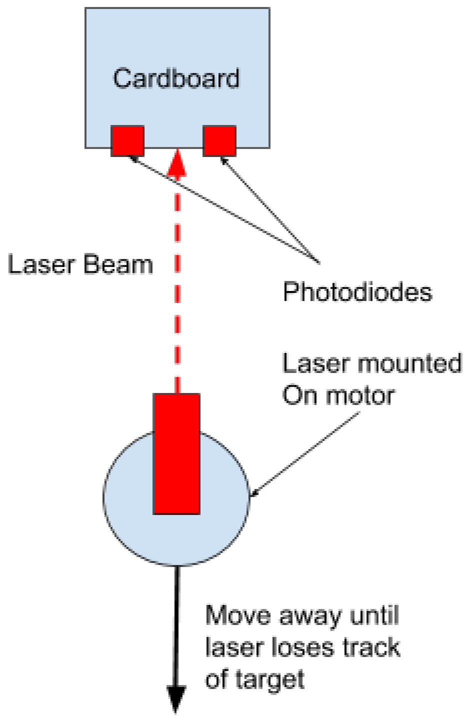
Testing the echo back, we sent the ascii character ‘U’(01010101) 1000x at varying baud rates, seeing at which baud rate would the communication start breaking down. We started at 9600 because it is a common communication rate. We found that communication falters at around 22000 baud. See appendix for more test baud rates.

Table 1. Percent of data corrupted compared to Baud rate of laser link. Laser and photodiode are 5 cm apart.

Baud Rate	Bytes Transferred	Bytes Corrupted	% Corrupted
9600	1000	0	0%
20000	1000	0	0%

21875	1000	0	0%
22000	1000	1000	100%
25000	1000	1000	100%

4.2. Tracking Test



For this test, the distance between the two photodiodes is 4 mm. According to the procedure, we start with the laser directly in front of the photodiodes and 1 cm away. We then move the laser and motor away from the receiver without moving laterally as shown in Figure 6. We record the maximum distance between the laser and receiver assembly where the laser loses track of the receiver.

The laser loses track of the receiver at a distance of ~20 cm.

With the equipment that we have, we could not devise other controlled tests to test the tracking ability. The reason for this is 1) no reliable way of measuring speed, 2) lack of equipment (such as a rail) to ensure that the laser and receiver remain on the same horizontal plane, and 3) breadboard and wires are cumbersome.

5. Conclusion

5.1. In our first test we successfully passed bits from an Arduino to itself. In future tests we will be passing bits from our ground station to the drone. This test shows that we have viable methods for optical wireless communication. In future tests we will have to mount the photodiode array on the drone, and improve our tracking solution. However having the optical wireless communications working is a fundamental part of our project which we have successfully implemented.

5.2. From the second test we take note that although we have successful tracking results in the azimuthal direction, we still have many areas of our tracking mechanism that require development. In future tests we will add onto the photodiode array techniques that we have developed in our second test. Some areas for future development include tracking in the elevation and azimuthal direction together and improving the tracking algorithms that tell the laser how much to shift when it hits an outer photodiode in the array.

6. Appendix

6.1. Echo test code

```
int b;

void setup() {

  pinMode(0,INPUT_PULLUP);
  pinMode(1,OUTPUT);
  Serial1.begin(10000);    //Increase until failure
}

void loop() {
  if (Serial1.available()) {
    Serial.write(Serial1.read());
  }
  if (Serial.available()) {
    Serial1.write(Serial.read());
  }
}
```

6.2. Echo Test Full Table

Baud Rate	# Bytes Transmitted	# Bytes Received	# Bytes Corrupted	% Corrupted/ Not Received
9600	1000	1000	0	0
20000	1000	1000	0	0
21000	1000	1000	0	0
21500	1000	1000	0	0
21750	1000	1000	0	0
21875	1000	1000	0	0
22000	1000	1000	1000	100
25000	1000	1000	1000	100
30000	1000	1000	1000	100
40000	1000	1000	1000	100

6.3. Tracking test code

```
#define IN1 8
#define IN2 9
#define IN3 10
#define IN4 11

#define LEFT   2
#define RIGHT  3

#define CHANGE_COUNT1 if(dirLeft) count1++; else count1 --;
volatile int count1 = 1;
volatile int step_count = 0;
int dirLeft = 1;
int delay_amt = 5;

void setup() {
  pinMode(IN1,OUTPUT);
  pinMode(IN2,OUTPUT);
  pinMode(IN3,OUTPUT);
  pinMode(IN4,OUTPUT);
  pinMode(LEFT,INPUT);
  pinMode(RIGHT,INPUT);
  attachInterrupt(digitalPinToInterrupt(LEFT), goRight, RISING);
  attachInterrupt(digitalPinToInterrupt(RIGHT), goLeft, RISING);
  Serial.begin(9600);
}

void loop() {
  delay(delay_amt);
  stepperISR();
  if (step_count > 1000){
    step_count = 0;
    dirLeft = !dirLeft;
  }
}

void out(int a, int b, int c, int d){
  digitalWrite(IN1,a);
  digitalWrite(IN2,b);
  digitalWrite(IN3,c);
  digitalWrite(IN4,d);
}

void stepperISR(){
  switch(count1){
    case 1:
      out(0,0,0,1);
      if (dirLeft) count1 ++;
      else count1 = 8;
      break;
    case 2:
      out(0,0,1,1);
      CHANGE_COUNT1
      break;
    case 3:
      out(0,0,1,0);
      CHANGE_COUNT1
      break;
    case 4:
      out(0,1,1,0);
      CHANGE_COUNT1
      break;
    case 5:
      out(0,1,0,0);
      CHANGE_COUNT1
      break;
    case 6:
      out(1,1,0,0);
      CHANGE_COUNT1
      break;
    case 7:
      out(1,0,0,0);
      CHANGE_COUNT1
      break;
    case 8:
      out(1,0,0,1);
      if (dirLeft) count1 = 1;
      else count1 --;
      break;
  }
  step_count ++;
}
```

```
void goLeft(){
  dirLeft = 1;
  step_count = 0;
}

void goRight(){
  dirLeft = 0;
  step_count = 0;
}
```