

A decorative graphic on the left side of the slide consisting of two overlapping parallelograms. The front one is blue and the back one is a light mint green. They are positioned diagonally, with the blue one partially covering the green one.

Autonomous Snowplow

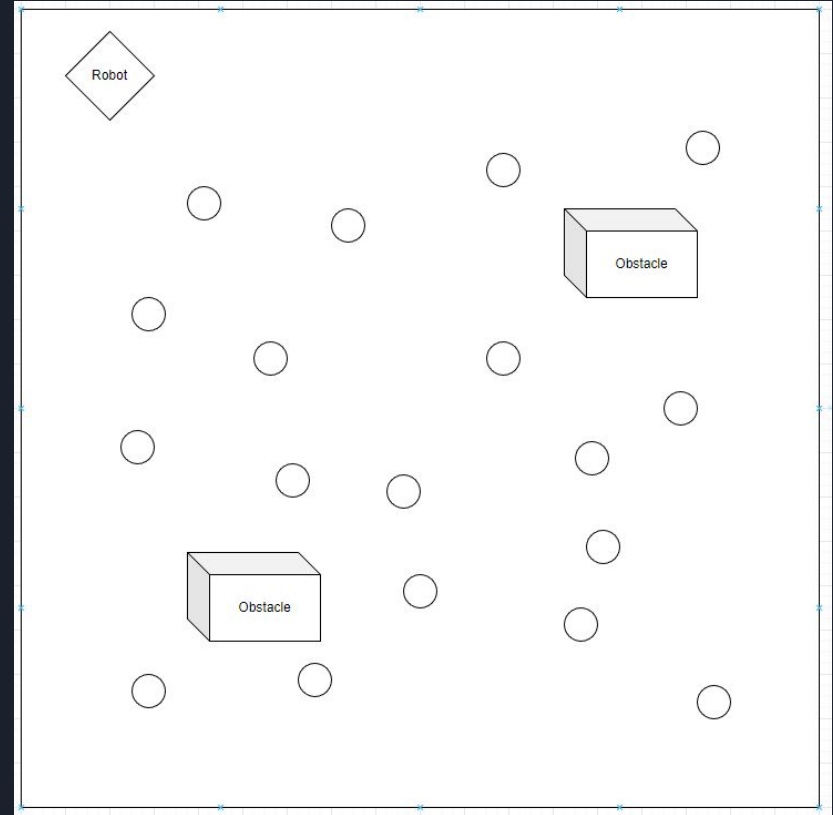
Team: Old Gold (L3-G4)
Wilson Amoussougbo & Tyler Mak

Snowplow

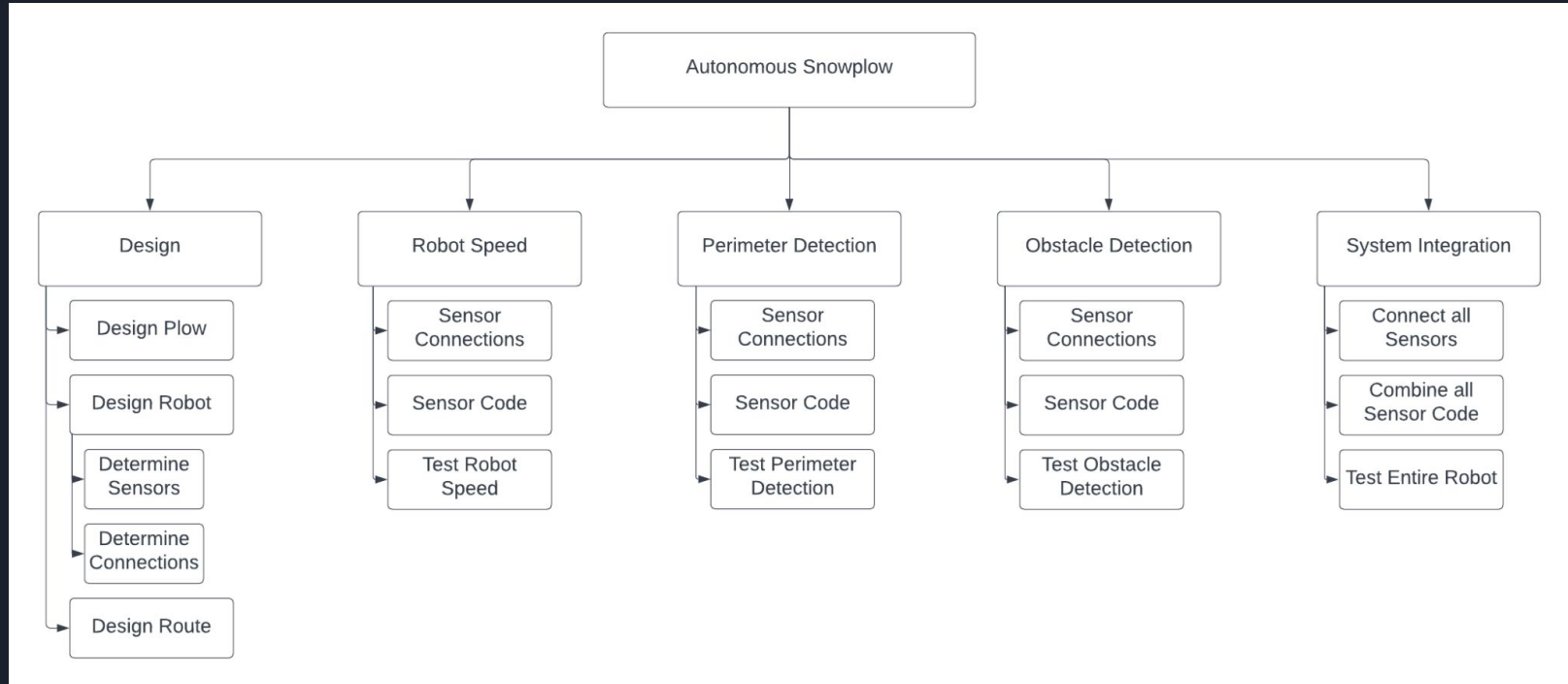


PROJECT STATEMENT

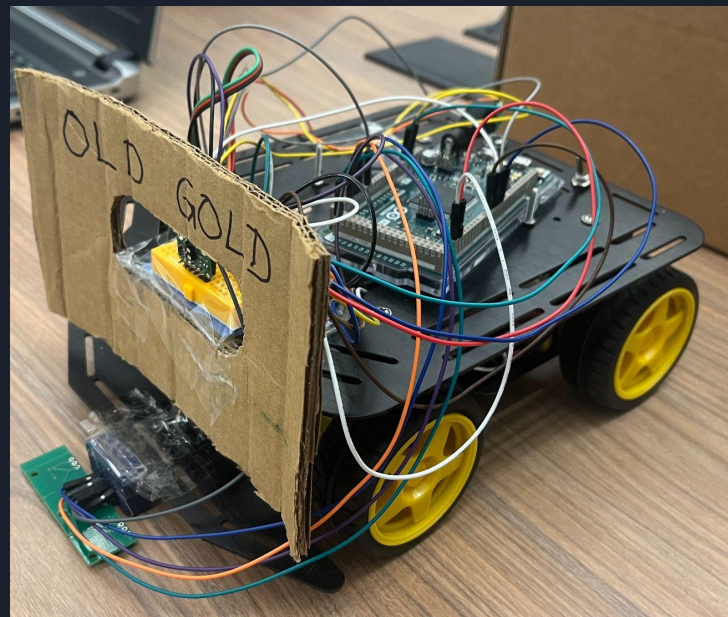
- *Robot can detect perimeter of arena*
- *Robot can detect and avoid obstacles*
- *Removes all the snowballs in the arena within 5 minutes*
- *Robot moves at a max speed of 30 cm/s*
- *Robot is a max size of 216 x 252 x 150 mm*



WORK BREAKDOWN STRUCTURE

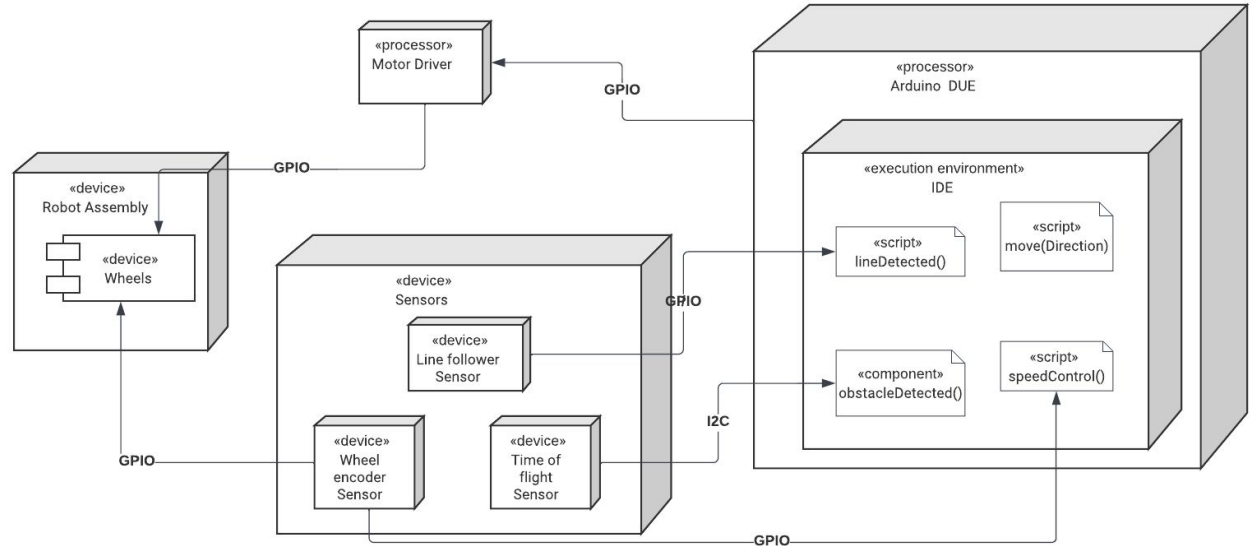


SOLUTION



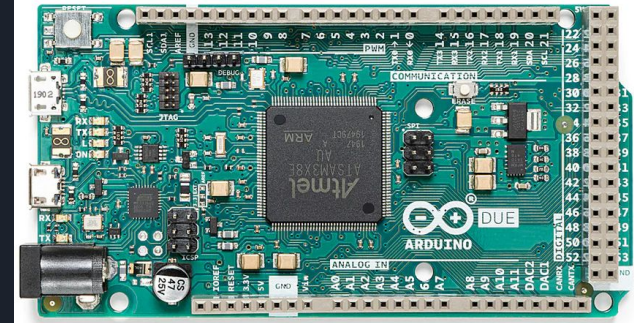
HARDWARE

- Arduino Due
- Motor Driver Board
- Line Follower Sensor
- Time of Flight Sensor
- Wheel Encoder Sensor



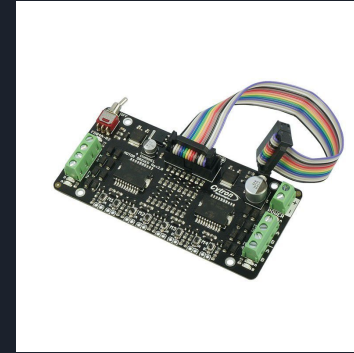
ARDUINO DUE

- Arduino Due
 - Microcontroller
 - Brain of the system and controls everything
 - Runs all our code



MOTOR DRIVER BOARD

- Motor Driver Board
 - Connects motors to Arduino
 - Controls direction and speed of each motor



```
//left_motor controls 2 wheels on left side of robot
//right_motor controls 2 wheels on right side of robot
#define left_motor_en 2
#define right_motor_en 3
#define left_motor_dir 4
#define right_motor_dir 5

//move robot forward using driver board
void forward(){
    printf("FORWARD\n");
    analogWrite(left_motor_en, 200);
    analogWrite(right_motor_en, 200);
    digitalWrite(left_motor_dir, LOW);
    digitalWrite(right_motor_dir, HIGH);
}

//move robot forward at slower speed
void slowForward(){
    printf("FORWARD SLOWER\n");
    analogWrite(left_motor_en, 150);
    analogWrite(right_motor_en, 150);
    digitalWrite(left_motor_dir, LOW);
    digitalWrite(right_motor_dir, HIGH);
}
```

```
//move robot backward
void backward(){
    printf("BACKWARD\n");
    analogWrite(left_motor_en, 255);
    analogWrite(right_motor_en, 255);
    digitalWrite(left_motor_dir, HIGH);
    digitalWrite(right_motor_dir, LOW);
}

//turn robot left
void left(){
    printf("LEFT\n");
    analogWrite(left_motor_en, 255);
    analogWrite(right_motor_en, 255);
    digitalWrite(left_motor_dir, HIGH);
    digitalWrite(right_motor_dir, HIGH);
    delay(1000); //turn left for 1s
}
```

```
//turn robot right
void right(){
    printf("RIGHT\n");
    analogWrite(left_motor_en, 255);
    analogWrite(right_motor_en, 255);
    digitalWrite(left_motor_dir, LOW);
    digitalWrite(right_motor_dir, LOW);
    delay(1000); //turn right for 1s
}

//stop robot movement
void stop(){
    printf("STOP\n");
    analogWrite(left_motor_en, 0);
    analogWrite(right_motor_en, 0);
    digitalWrite(left_motor_dir, LOW);
    digitalWrite(right_motor_dir, LOW);
}
```


LINE FOLLOWER SENSOR

- Used to detect the perimeter (black tape)
- Consists of 3 sensors
- Attached to front of robot facing the ground
- Black tape detected if at least 1 of the 3 sensors detect black



```
#include "lineDetector.h"

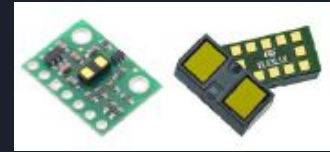
#define line_detector_front1 A9
#define line_detector_front2 A10
#define line_detector_front3 A11

//method for perimeter detection using line sensor
bool checkForLine()
{
    //read from each sensor
    int ex1 = analogRead(line_detector_front1);
    int ex2 = analogRead(line_detector_front2);
    int ex3 = analogRead(line_detector_front3);

    //print readings
    printf("Left analog: %i Middle analog: %i Right analog: %i\n", ex1, ex3, ex2);
    //return true if black detected for any sensor false otherwise
    if (ex1 > 900 || ex2 > 950 || ex3 > 900) //different tested threshold for all sensors
        return true;
    else
        return false;
}
```

TIME OF FLIGHT SENSOR

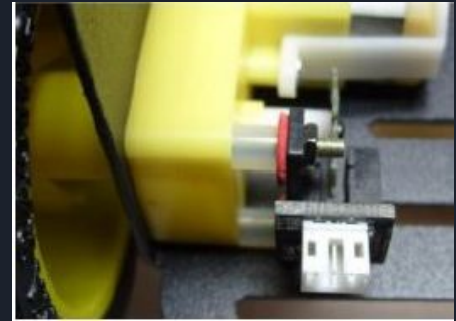
- Used as a distance sensor for detecting obstacles
- Makes use of I2C protocol to communicate with Arduino
- Attached at the front at the top of our robot
- Detects obstacle if detected at least 20 cm in front of the robot



```
//ToF sensor
sensorToF.setTimeout(500); //0.5s timeout
if (!sensorToF.init()){
  Serial.println("Failed to detect and initialize ToF sensor.");
  while(1);
}
sensorToF.setDistanceMode(VL53L1X::Short); //ToF in short mode
sensorToF.setMeasurementTimingBudget(20000);
sensorToF.startContinuous(50);
```

```
//method for obstacle detection using ToF sensor
bool checkForObstacle(){
  sensorToF.read();
  Serial.println(sensorToF.ranging_data.range_mm);
  if (sensorToF.ranging_data.range_mm <= 200) //if obstacle detected within 200mm of robot
    return true;
  else
    return false;
}
```

WHEEL ENCODER SENSOR



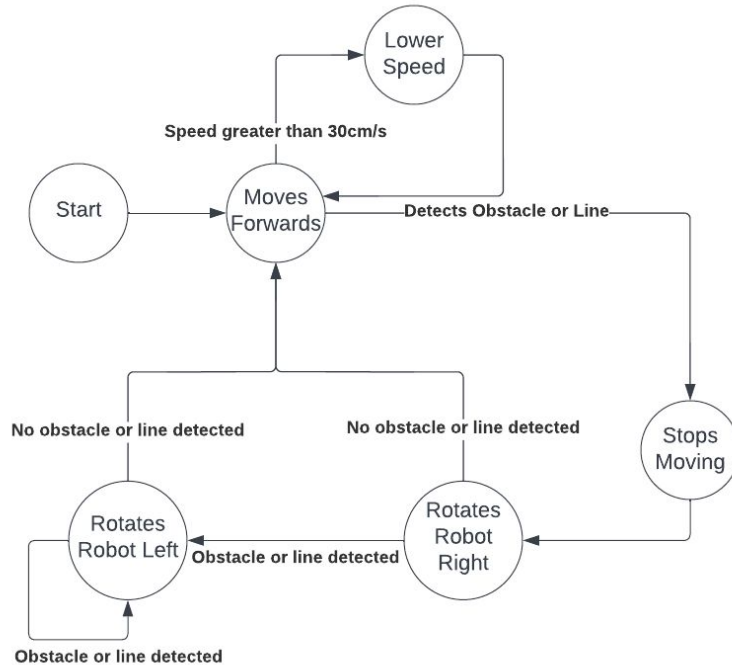
- Used to determine the speed of the robot
- Attached to the front wheels
- Ensures robot is going at a specified speed and doesn't exceed the limit
- Sends control flag if robot is going above 30 cm/s

```
//Wheel encoder
PMC->PMC_PCER0 |= PMC_PCER0_PID28; // Timer Counter 0 channel 1 IS TC1, TC1 power ON
TC0->TC_CHANNEL[1].TC_CMR = TC_CMR_TCCLKS_TIMER_CLOCK1 // capture mode, MCK/2 = 42 MHz
| TC_CMR_ABETRIG // TIOA is used as the external trigger
| TC_CMR_LDRA_RISING // load RA on rising edge of TIOA
| TC_CMR_ETRGEDG_RISING; // Trigger on rising edge
TC0->TC_CHANNEL[1].TC_CCR = TC_CCR_SWTRG | TC_CCR_CLKEN; // Reset TC counter and enable
TC0->TC_CHANNEL[1].TC_IER |= TC_IER_LDRAS; // Trigger interrupt on Load RA
NVIC_EnableIRQ(TC1_IRQn); // Enable TC1 interrupts
}
```

```
if (CaptureFlag) { //wheel encoder interrupt
    CaptureFlag = 0; //Reset the flag,
    speed = (19.648/(CaptureCountA/42000.0/1000.0)/10.0); //calculate speed
    printf("Speed: %f cm/s\n", speed);
    if (speed > 30){
        speedFlag = true; //robot going too fast
    } else {
        speedFlag = false; //robot moving below max allowed speed
    }
}
```

```
void TC1_Handler() { //interrupt handler
    uint32_t status = TC0->TC_CHANNEL[1].TC_SR; //Read status register, Clear status
    if (status & TC_SR_LDRAS) { // If ISR is fired by LDRAS then ....
        CaptureCountA = TC0->TC_CHANNEL[1].TC_RA; //read TC_RA
        speed = (19.648/(CaptureCountA/42000.0/1000.0)/10.0);
        if (speed < 40) { //debouncing any extremely high speed reports
            CaptureFlag = 1; //Inform the main loop of an update
        }
    }
}
```

MOVEMENT ALGORITHM



```
switch(state){ //state machine
case 0: //move forward state
    if (speedFlag){ //check if robot moving too fast
        slowForward();
    } else {
        forward();
    }
    if (line || obstacle){ //check for perimeter or obstacle
        stop();
        state = 1; //turn right
    } else {
        state = 0; //continue forward
    }
    break;
```

```
case 1: //turn right state
    right();
    if (line || obstacle){
        stop();
        state = 2; //turn left
    } else {
        state = 0; //continue forward
    }
    break;
case 2: //turn left state
    left();
    if (line || obstacle){
        stop();
        state = 2; //turn left again
    } else {
        state = 0; //continue forward
    }
    break;
}
delay(500); //poll every 0.5s
```

TESTING RESULTS



- Demonstration Expectations
- Demonstration Results
- Resolutions



Questions?