```
// PID Constants
#define kp 25
#define ki 0.8
#define kd 5


task main() {
        // the light sensor value
        int sensor;
        int sensorLeft, sensorRight;

        // the offset for zero level
        int offset = 0;

        // the target power
        int tp = 0;

        // turnration and error value
        int turn, error;

        // PID variables
        int integral = 0, lastError = 0, derivate = 0;

        int dt = 5;

//      Off(OUT_A);


        // Initialize touch sensor for software control
        SetSensorTouch(S1);
        ResetSensor(S1);
        SetSensorMode(S1, SENSOR_MODE_EDGE);
        SetSensorTouch(S4);
        ResetSensor(S4);
        SetSensorMode(S4, SENSOR_MODE_EDGE);

        // Initialize light sensor for attitude control
  //SetSensor(IN_3, SENSOR_LIGHT);
        SetSensorLight(IN_3);
  SetSensor(IN_3, SENSOR_COLORRED);
  //SetSensorType(IN_3, SENSOR_TYPE_LIGHT_ACTIVE);
  //SetSensorType(IN_3, IN_TYPE_COLORRED);
  //SetSensorMode(IN_3, SENSOR_MODE_EDGE);
        SetSensorMode(IN_3, SENSOR_MODE_RAW);
  //SetSensorMode(IN_3, SENSOR_MODE_PERCENT);
        //ResetSensor(IN_3);

        while (Sensor(IN_1) == 0) {
                TextOut(0, LCD_LINE3, "Waiting for start", false);
                TextOut(0, LCD_LINE2, "Push bazooka", false);
                Wait(100);
                ClearScreen();
```

```
     }

     while (Sensor(IN_1) == 1) {
             offset = Sensor(IN_3)/10;
             TextOut(0, LCD_LINE3, "Accquiring offset...", true);
             Wait(100);
     }

     while (Sensor(IN_1) != 3) {
 ClearScreen();
       TextOut(0, LCD_LINE3, "Balancing...", false);

             sensor = Sensor(IN_3)/10;
             sensorRight = Sensor(IN_1) % 2;
             sensorLeft = Sensor(IN_4) % 2;

             error = sensor - offset;
             integral = integral + error;
             derivate = (error - lastError);

             turn = ((kp * error) + (ki * integral) + (kd * derivate));

lastError = error;

NumOut(0, LCD_LINE1, sensor, false);
NumOut(0, LCD_LINE5, sensorLeft, false);
NumOut(5, LCD_LINE5, sensorRight, false);

             if (turn > 0) {
        /*if(sensorLeft == 1) {
    OnFwd(OUT_B, turn);
    OnFwd(OUT_C, turn/1.5);

    //OnFwd(OUT_C, turn);
    //RotateMotor(OUT_B, 1, 0.1);

  } else if(sensorRight == 1) {
    OnFwd(OUT_B, turn/1.5);
    OnFwd(OUT_C, turn);

    //OnFwd(OUT_B, turn);
    //RotateMotor(OUT_C, 1, 0.1);
  } else {
                       OnFwd(OUT_BC, turn);
  } */

  OnFwd(OUT_AB, turn);

             } else if (turn < 0) {
        /*if(sensorLeft == 1) {
    OnRev(OUT_B, -(turn/1.5));
    OnRev(OUT_C, -(turn));
```

```
        //OnRev(OUT_B, -turn);
        //RotateMotor(OUT_C, 1, -0.1);

      } else if(sensorRight == 1) {
        OnRev(OUT_B, -(turn));
        OnRev(OUT_C, -(turn/1.5));

        //OnRev(OUT_C, -turn);
        //RotateMotor(OUT_B, 1, -0.1);

      } else {
                        OnRev(OUT_BC, -turn);
      } */

      OnRev(OUT_AB, -turn);

                } else {
                        Off(OUT_AB);
                }

/*    if(remotecount % 1000 == 10){
      RotateMotor(OUT_A, 10, 5);
      RotateMotor(OUT_A, 10, -6);
    } else if ( remotecount % 500 == 0){
      RotateMotor(OUT_A, 20, -10);
                } else {
                        Off(OUT_A);
                } */

                Wait(dt);
        }
}
```