Design note: Nanobot

Adrie Huesman
Start date: Jan 2022
Last update: 19 June 2023

1. Introduction

Nanobot was designed to be a simple, general purpose, small (desktop) and cost-effective robot. The robot should support testing of sensors, control approaches and wireless communications. In the end the following functionality was implemented using Nanobot as a platform:

1. Line following via InfraRed (IR) reflective sensor
2. Sumo (push objects out of the ring) with IR and Ultrasonic (US) sensor
3. Line following via Huskylens
4. Obstacle avoidance with US sensor mounted on servo and IR proximity sensors
5. Light seeking with two LDR sensors
6. Bluetooth Remote Control (RC) (full proportional control)

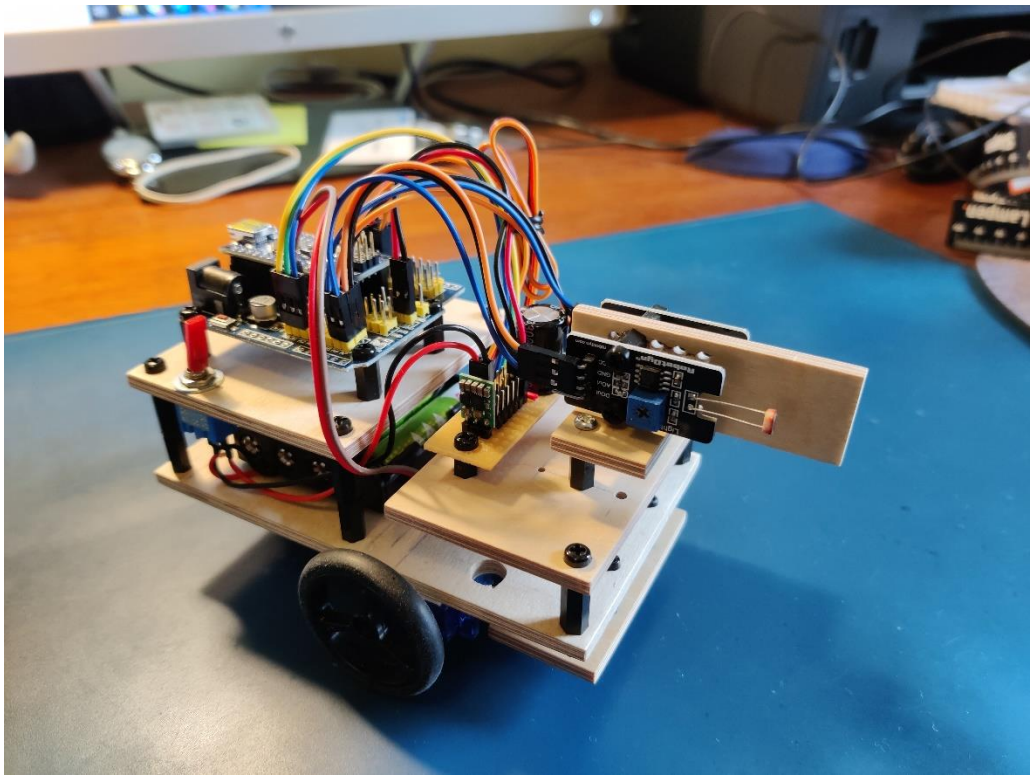

Figure 1: Nanobot with the light seeking sensor attached to it.

2. Design details

2.1 Drive

A 2-wheel differential drive with coaster ball was selected. The motors are SG90 continuous (360 degrees) servos. This gives adequate performance and is very cost-effective. It also eliminates the need for a motor driver.

2.2 Chassis

The chassis was made of 4 mm aviation plywood. It measures 8 by 13 cm and there are essentially two levels. The lower level is reserved for the motors and the batteries, the upper level is for the controller and sensors.

2.3 Power

The power comes from 4 AAA batteries which via a power switch is directed to a self-constructed power board. This board contains a 5-volt step-up step-down regulator (S7V7F5 Pololu 2119) and a large capacitor (3300 microfarad). This means that the robot can run on either alkaline or rechargeable NiMH batteries. The capacitor allows for the servos to run off the stabilized 5-volt eliminating a varying voltage supply as a possible disturbance.

2.4 Controller

As controller an Arduino Nano was chosen. It is adequate, small, cost-effective and supports a lot of libraries! The facilitate connections the Arduino Nano was placed on an Arduino Nano sensor I/O shield v3.0. The shield offers the extra advantage that the Arduino Nano can be attached to the chassis via M3 spacers.

2.5 Sensors

As standard sensors a QTR-3A Reflectance Sensor Array (Pololu 2456) and an Ultrasonic Distance Sensor - HC-SR04 mounted on a 270-degree servo were added. These sensors support functionality like line tracking and obstacle avoidance. Also, a potentiometer was added to trim the difference in left and right motor speed. This is strictly speaking not a sensor but proved to be very useful. The potentiometer can also be used to reduce battery voltage below 5 volts so it can be measured by the controller.

2.6 Software

The software was written/developed in the Arduino IDE. From a control point of view the software are implementations of proportional control and/or a Sequential Function Chart (SFC). To facilitate using the continuous servos two functions were written (making use of `Servo.h`):

```
// Set speed of left motor, -500 < speed < 500
void Lspeed(int speed)
{
  Lservo.writeMicroseconds(speed + 1500);
}

// Set speed of right motor, -500 < speed < 500
```

```
void Rspeed(int speed)
{
  Rservo.writeMicroseconds(1500 - speed);
}
```

So, the speed can be set by an integer between -500 and 500. Negative numbers mean driving backwards. Note the difference in sign inside Lspeed and Rspeed.

Figure 2 shows the SFC used for the Sumo implementation. Figure 3 gives the SFC for obstacle avoidance.

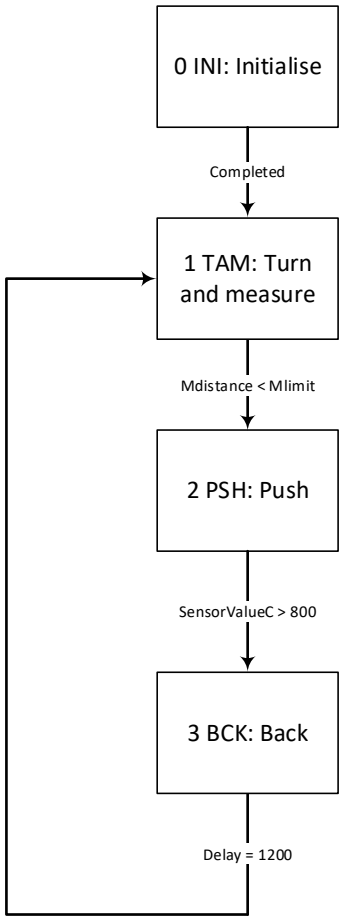

Figure 2: The SFC for the Sumo functionality.

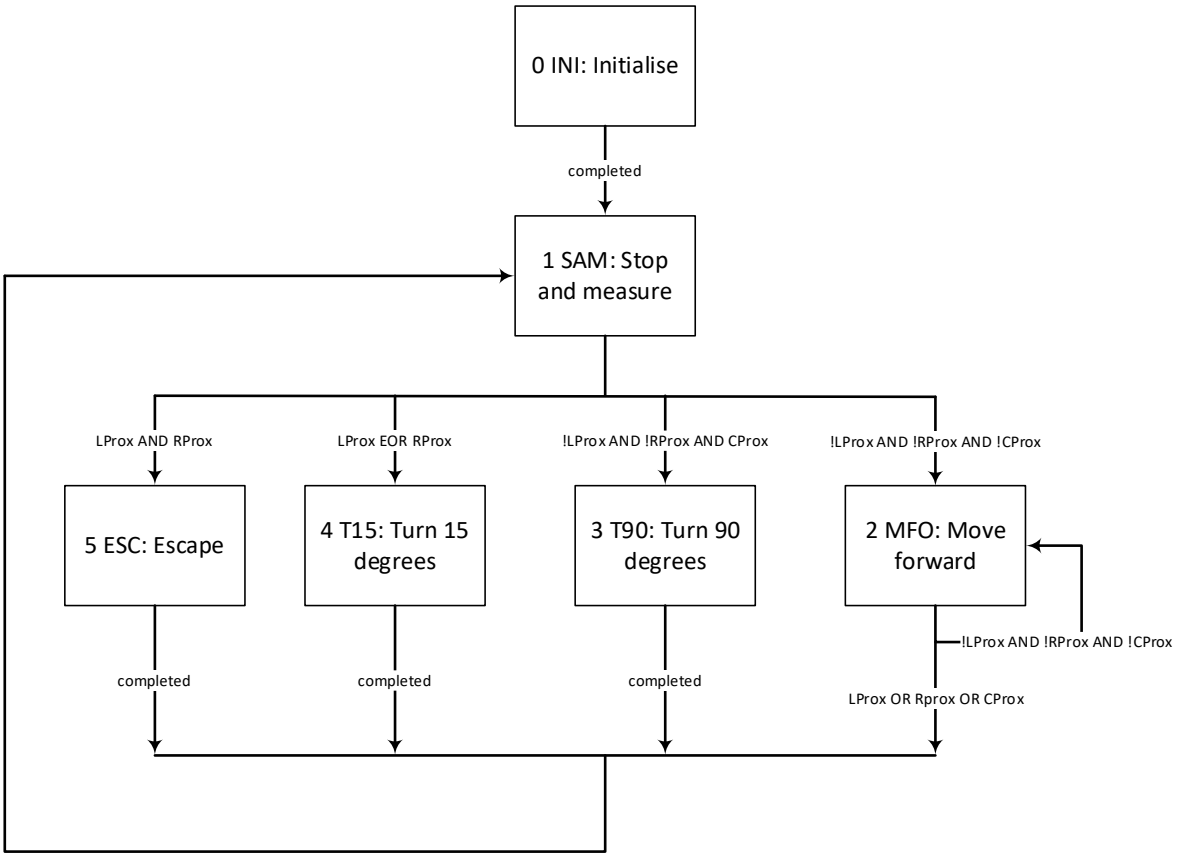

Figure 3: The SFC for obstacle avoidance.

To help the robot going straight (in the state MFO) the potentiometer connected to analog input A7 can be adjusted such that both continuous servo motors operate at the same speed. Although this can be achieved by software alone it is much faster and easier to do this with a potentiometer. Furthermore, adjustments can be made without having to change the sketch! Note that the measured voltage (0 - 1023) is mapped to a bias (-100 – 100). The bias is added/subtracted in such a way that turning the potentiometer to the right implies that the robot increases its tendency to turn right.

The table below was used to implement the proportional RC functionality (together with Arduino Bluetooth Controlled Joystick App). Please note that in the quadrant between 0 and 90 degrees the left motor operates constantly at +Mspeed while the right motor varies from -Mspeed to +Mspeed.

Table 1: The motor speed as function of the angle.

Angle*	**Left**	**Right**	**Comment**
0	+Mspeed	-Mspeed	Rotate CW
90	+Mspeed	+Mspeed	Move forward
180	-Mspeed	+Mspeed	Rotate CCW
270	-Mspeed	-Mspeed	Move back
360	+Mspeed	-Mspeed	Equal to 0

*Angles are measured from the x-axis CCW.

The appendices A – F give the Arduino sketches of the various functionalities. Note that all sketches have quite a bit of useful comment at the top.

3. Performance

The performance of Nanobot is surprisingly good for such a simple robot. It proved to be a good test bed for every functionality mentioned above. Its versatility also makes it a lot of fun.

4. Conclusions and recommendations

- Use of an Arduino Nano and Nano IO shield is convenient. Sensor and servos can be connected very easily. Furthermore, compared to an Arduino Uno a Nano has two extra (analog) pins. Just as the Uno it is based on an ATmega328P microcontroller so excellent library support.
- Having a step-up/step-down converter working at 5 volts is a good idea:
    - The 5 volts can be used to drive all electronics. This includes the servo motors and therefore eliminates supply voltage drop as a possible disturbance.
    - The 5 volts can be generated efficiently by 4 or 5 NiMH or 4 alkaline batteries.
- Consider using 3 mm plywood better to use with 2 and 3 mm spacers.
- Note that Pololu ball casters should be oriented.
- For cables to pass through the chassis drill 10 mm with a sharp wood drill so a 3 pin Dupont connector can pass easily.

The Nano platform is small (suited for a table), convenient (easy to connect things) and low cost (see table below). However, it does have limitations, the most important one being that continuous servo motors cannot be fitted with encoders. So, closed-loop speed and/or position control are not possible.

Table 2: An overview of the costs (€ in 2022).

Item	**Price**
Continuous servo SG90	3,75
Continuous servo SG90	3,75
Wheel for Standard Servo 25T-40×7mm 2-Pack Pololu 4905	2,80
Ball Caster with 3/4” Plastic Ball Pololu 954	3,50
QTR-3A Reflectance Sensor Array Pololu 2456	4,05
4-AAA battery holder Pololu 1145	1,55
Power switch	0,60
Arduino Nano (copy)	7,85
Arduino Nano sensor I/O shield v3.0	4,45
5V Step-Up/Step-Down Voltage Regulator S7V7F5 Pololu 2119	10,45
Servo 270 degrees TS90A	3,75
US sensor HC-SR04	2,15
M3 plastic standoffs, nuts, bolts, cables etc.	0,50
Chassis (plywood)	2,00
Total	**51,15**

Appendix A: Line following via IR reflectance sensor

```
/*
NanoBot
Adrie Huesman
13 September 2021

Hardware
Arduino Nano Atmega328
Arduino Nano sensor I/O shield v3.0
4 Alkaline batteries so 6 volts
2 SG90 continous or 360 degrees servos
Wheel for Standard Servo 25T-40×7mm 2-Pack Pololu 4905
Ball Caster with 3/4" Plastic Ball Pololu 954
QTR-3A Reflectance Sensor Array Pololu 2456
Powerboard, own design based on 5V Step-Up/Step-Down Voltage Regulator S7V7F5 Pololu 2119
SG90 270 degrees servo and ultrasonic distance sensor HC-SR04
Chassis, own design 4mm aviation plywood 13 by 8 cm (two levels)

Software

I/O
D0 --
D1 --
D2 --
D3 --
D4 --
D5 Continuous servo left
D6 Continuous servo right
D7 --
D8 --
D9 --
D10 --
D11 --
D12 --
D13 --
A0 Sensor array left
A1 Sensor array center
A2 Sensor array right
A3 --
A4 --
A5 --
A6 --
A7 --

Other
SensorValues when all black 900 - 924, say 900
SensorValues when all white 488 - 579, say 600
 */

// Declare variables
int sensorValueL;     // variable to store the value coming from the left sensor
int sensorValueC;     // variable to store the value coming from the center sensor
int sensorValueR;     // variable to store the value coming from the right sensor
int Error;
float Kc = 0.40;      // Controller gain
float Paction;
int Avspeed = 150;    // Average speed

#include <Servo.h>
Servo Lservo;
Servo Rservo;

// Set speed of left motor, -500 < speed < 500
void Lspeed(int speed)
{
  Lservo.writeMicroseconds(speed + 1500);
}

// Set speed of right motor, -500 < speed < 500
void Rspeed(int speed)
```

```
{
  Rservo.writeMicroseconds(1500 - speed);
}

void setup()
{
  Serial.begin(9600);   // set up Serial library at 9600 bps
  Lservo.attach(5);
  Rservo.attach(6);
}

void loop()
{
  sensorValueL = analogRead(A0);
  sensorValueC = analogRead(A1);
  sensorValueR = analogRead(A2);
  Error = sensorValueL - sensorValueR;
  Paction = Kc*Error;
  Lspeed(Avspeed - int(Paction));
  Rspeed(Avspeed + int(Paction));
  delay(10);
}
```

Appendix B: Sumo (push objects out of the ring)

```
/*
SumoB
Adrie Huesman
28 December 2021

Hardware
Arduino Nano Atmega328
Arduino Nano sensor I/O shield v3.0
4 Alkaline batteries so 6 volts
2 SG90 continous or 360 degrees servos
Wheel for Standard Servo 25T-40×7mm 2-Pack Pololu 4905
Ball Caster with 3/4" Plastic Ball Pololu 954
QTR-3A Reflectance Sensor Array Pololu 2456
Powerboard, own design based on 5V Step-Up/Step-Down Voltage Regulator S7V7F5 Pololu 2119
SG90 270 degrees servo and ultrasonic distance sensor HC-SR04
Chassis, own design 4mm aviation plywood 13 by 8 cm (two levels)

Software

I/O
D0 -- Reserved UART Rx
D1 -- Reserved UART Tx
D2 --
D3 --
D4 --
D5 Continuous servo left
D6 Continuous servo right
D7 --
D8 --
D9 270 degrees servo
D10 --
D11 --
D12 HC-SR04 Trig
D13 HC-SR04 Echo
A0 Sensor array left
A1 Sensor array center
A2 Sensor array right
A3 --
A4 -- Reserved I2C SDA
A5 -- Reserved I2C SDL
A6 --
A7 --

Other
SensorValues when all black 900 - 924, say 900
SensorValues when all white 488 - 579, say 600

Same RPMs achieved when:
Forward Lspeed 200 Rspeed 240
Backward Lspeed -250 Rspeed -200
Left turn Lspeed -200 Rspeed 200
Right turn Lspeed 200 Rspeed -200
 */

// Declare variables
int sensorValueL;     // variable to store the value coming from the left sensor
int sensorValueC;     // variable to store the value coming from the center sensor
int sensorValueR;     // variable to store the value coming from the right sensor
int Avspeed = 200;    // Average speed (200)
enum state
{
  INI,
  TAM,
  PSH,
  BCK
};
state Robotstate = INI;
float Mdistance;
float Mlimit = 15.0;
```

```
#include <Servo.h>
#include <HCSR04.h>
Servo Lservo;
Servo Rservo;

// Initialize sensor that uses digital pins 12 and 13
const byte triggerPin = 12;
const byte echoPin = 13;
UltraSonicDistanceSensor distanceSensor(triggerPin, echoPin);

// Set speed of left motor, -500 < speed < 500
void Lspeed(int speed)
{
  Lservo.writeMicroseconds(speed + 1500);
}

// Set speed of right motor, -500 < speed < 500
void Rspeed(int speed)
{
  Rservo.writeMicroseconds(1500 - speed);
}

void setup()
{
  Serial.begin(9600);   // set up Serial library at 9600 bps
  Lservo.attach(5);
  Rservo.attach(6);
}

void loop()
{
  switch(Robotstate)

  {
    case INI:
    Lspeed(0);
    Rspeed(0);
    delay(100);
    Mdistance = distanceSensor.measureDistanceCm();
    delay(100);
    Robotstate = TAM;
    break;

    case TAM:
    Lspeed(200);
    Rspeed(-200);
    Mdistance = distanceSensor.measureDistanceCm();
    while (Mdistance > Mlimit)
    {
      Mdistance = distanceSensor.measureDistanceCm();
      // delay(10);
    }
    Lspeed(0);
    Rspeed(0);
    Robotstate = PSH;
    break;

    case PSH:
    Lspeed(200);
    Rspeed(240);
    sensorValueC = analogRead(A1);
    while (sensorValueC < 800)
    {
      sensorValueC = analogRead(A1);
      delay(10);
    }
    delay(200); // to push the object really outside the circle
    Lspeed(0);
    Rspeed(0);
    Robotstate = BCK;
```

```
    break;

    case BCK:
    Lspeed(-250);
    Rspeed(-200);
    delay(1200); // to return to the center of the cirle
    Lspeed(0);
    Rspeed(0);
    Robotstate = TAM;
    break;
  }
}
```

Appendix C: Line following via Huskylens

```
/*
HuskyLineTracking
Adrie Huesman
20 Januari 2021

Hardware
Arduino Nano Atmega328
Arduino Nano sensor I/O shield v3.0
4 Alkaline batteries so 6 volts
2 SG90 continous or 360 degrees servos
Wheel for Standard Servo 25T-40×7mm 2-Pack Pololu 4905
Ball Caster with 3/4" Plastic Ball Pololu 954
QTR-3A Reflectance Sensor Array Pololu 2456
Powerboard, own design based on 5V Step-Up/Step-Down Voltage Regulator S7V7F5 Pololu 2119
GRAVITY: HUSKYLENS - 2MP also see https://wiki.dfrobot.com/HUSKYLENS_V1.0_SKU_SEN0305_SEN0336
Chassis, own design 4mm aviation plywood 13 by 8 cm (two levels)

Software
I/O
D0 --
D1 --
D2 --
D3 --
D4 --
D5 Continuous servo left
D6 Continuous servo right
D7 --
D8 --
D9 --
D10 --
D11 --
D12 --
D13 --
A0 Sensor array left
A1 Sensor array center
A2 Sensor array right
A3 --
A4 I2C SDA Huskylens
A5 I2C SCL Huskylens
A6 --
A7 --

Other
SensorValues when all black 900 - 924, say 900
SensorValues when all white 488 - 579, say 600
 */

#include "HUSKYLENS.h"
#include "SoftwareSerial.h"
#include <Servo.h>

HUSKYLENS huskylens;
//HUSKYLENS green line >> SDA; blue line >> SCL
void printResult(HUSKYLENSResult result);

Servo Lservo;
Servo Rservo;

// Declare variables
int32_t error;
int ID1;
float Kc = 0.80;      // Controller gain (0.80)
float Paction;
int Avspeed = 200;    // Average speed (200)

void printResult(HUSKYLENSResult result){
    if (result.command == COMMAND_RETURN_BLOCK){
```

```
Serial.println(String()+F("Block:xCenter=")+result.xCenter+F(",yCenter=")+result.yCenter+F(",w
idth=")+result.width+F(",height=")+result.height+F(",ID=")+result.ID);
    }
    else if (result.command == COMMAND_RETURN_ARROW){

Serial.println(String()+F("Arrow:xOrigin=")+result.xOrigin+F(",yOrigin=")+result.yOrigin+F(",x
Target=")+result.xTarget+F(",yTarget=")+result.yTarget+F(",ID=")+result.ID);
    }
    else{
        Serial.println("Object unknown!");
    }
}

void Lspeed(int speed)   // Set speed of left motor, -500 < speed < 500
{
  Lservo.writeMicroseconds(speed + 1500);
}

void Rspeed(int speed)   // Set speed of right motor, -500 < speed < 500
{
  Rservo.writeMicroseconds(1500 - speed);
}

void setup() {
    Lservo.attach(5);
    Rservo.attach(6);
    Serial.begin(115200);
    Wire.begin();
    while (!huskylens.begin(Wire))
    {
        Serial.println(F("Begin failed!"));
        Serial.println(F("1.Please recheck the \"Protocol Type\" in HUSKYLENS (General
Settings>>Protol Type>>I2C)"));
        Serial.println(F("2.Please recheck the connection."));
        delay(100);
    }
    huskylens.writeAlgorithm(ALGORITHM_LINE_TRACKING); //Switch the algorithm to line
tracking.
}

void loop() {
    if (!huskylens.request()) Serial.println(F("Fail to request data from HUSKYLENS, recheck
the connection!"));
    else if(!huskylens.isLearned()) Serial.println(F("Nothing learned, press learn button on
HUSKYLENS to learn one!"));
    else if(!huskylens.available()) Serial.println(F("No block or arrow appears on the
screen!"));
    else
    {
        HUSKYLENSResult result = huskylens.read();
        printResult(result);

        // Calculate the error
        error = (int32_t)160 - (int32_t)result.xTarget;

        // Perform P-only control
        Paction = Kc*error;
        Lspeed(Avspeed - int(Paction));
        Rspeed(Avspeed + int(Paction) + 40);  // Motors behave different
        delay(10);
    }
}
```

Appendix D: Obstacle avoidance

```
/*
NanoRoverC
Adrie Huesman
Start: 26 January 2022
Last update: 26 Feb 2022

Hardware
Arduino Nano Atmega328
Arduino Nano sensor I/O shield v3.0
4 Alkaline batteries so 6 volts
2 SG90 continous or 360 degrees servos
Wheel for Standard Servo 25T-40×7mm 2-Pack Pololu 4905
Ball Caster with 3/4" Plastic Ball Pololu 954
QTR-3A Reflectance Sensor Array Pololu 2456
Powerboard, own design based on 5V Step-Up/Step-Down Voltage Regulator S7V7F5 Pololu 2119
SG90 270 degrees servo and ultrasonic (US) distance sensor HC-SR04
2 Digital infrared proximity sensors FC-51 or HW-201
Chassis, own design 4mm aviation plywood 13 by 8 cm (two levels)

Software

I/O
D0 -- Reserved UART Rx
D1 -- Reserved UART Tx
D2 Right proximity sensor
D3 Left proximity sensor
D4 --
D5 Continuous servo left
D6 Continuous servo right
D7 --
D8 --
D9 270 degrees servo
D10 --
D11 --
D12 HC-SR04 Trig
D13 HC-SR04 Echo
A0 Sensor array left
A1 Sensor array center
A2 Sensor array right
A3 --
A4 -- Reserved I2C SDA
A5 -- Reserved I2C SDL
A6 --
A7 Potmeter to set bias between motor speeds

Other
SensorValues when all black 900 - 924, say 900
SensorValues when all white 488 - 579, say 600

Same RPMs achieved when:
Forward Lspeed 200 Rspeed 240
Backward Lspeed -250 Rspeed -200
Left turn Lspeed -200 Rspeed 200
Right turn Lspeed 200 Rspeed -200
*/

// Declare variables
int Avspeed = 200;   // Only used in MFO state, typical value 200
int val;   // Value from potentiometer
int bias;   // To run left and right motor at same speed (33 - 34)
enum state
{
  INI,
  SAM,
  MFO,
  T90,
  T15,
  ESC
};
```

```
state Robotstate = INI;
float Cdistance;
float Climit = 15.0;
float Ldistance;
float Rdistance;
bool LProx;
bool RProx;
bool CProx;

#include <Servo.h>
#include <HCSR04.h>
Servo Lservo;
Servo Rservo;
Servo Cservo;

// Initialize US sensor that uses digital pins 12 and 13
UltraSonicDistanceSensor distanceSensor(12, 13);   // Trigger and echo

// Set speed of left motor, -500 < speed < 500
void Lspeed(int speed)
{
  Lservo.writeMicroseconds(speed + 1500);
}

// Set speed of right motor, -500 < speed < 500
void Rspeed(int speed)
{
  Rservo.writeMicroseconds(1500 - speed);
}

// Exclusive OR via truth table, see below
// An alternative is to use (a || b) && !(a && b)
bool EOR(bool a, bool b)
{
  bool c;
  if (!a && !b) {c = false;}
  if (a && !b) {c = true;}
  if (!a && b) {c = true;}
  if (a && b) {c = false;}
  return c;
}

void setup()
{
  pinMode(2, INPUT);    // Sets digital pin 2 as input for RProx
  pinMode(3, INPUT);    // sets digital pin 3 as input for LProx
  Serial.begin(9600);   // Set up Serial library at 9600 bps
  Lservo.attach(5);
  Rservo.attach(6);
  Cservo.attach(9);
  Cservo.writeMicroseconds(1430);   // set HC-SR04 at 90 degrees
  delay(500);
  Cservo.writeMicroseconds(2110);   // set HC-SR04 at 0 degrees
  delay(500);
  Cservo.writeMicroseconds(790);   // set HC-SR04 at 180 degrees
  delay(500);
  Cservo.writeMicroseconds(1430);   // set HC-SR04 at 90 degrees
  delay(500);
}

void loop()
{
  switch(Robotstate)
  {
    case INI:
    Lspeed(0);
    Rspeed(0);
    Cservo.writeMicroseconds(1430);
    delay(100);
    Cdistance = distanceSensor.measureDistanceCm();
    delay(100);
```

```
    Robotstate = SAM;
    break;

    case SAM:
    Lspeed(0);
    Rspeed(0);
    Cservo.writeMicroseconds(1430);
    Cdistance = distanceSensor.measureDistanceCm();
    LProx = !(digitalRead(3));   // LProx is true when an object is detected
    RProx = !(digitalRead(2));   // RProx is true when an object is detected
    CProx = (Cdistance < Climit);   // CProx is true when an object is detected
    if (EOR(LProx, RProx))   // Object detected left or right
    {
      Robotstate = T15;
    }
    if (!LProx && !RProx && CProx)   // No object detected left or right, but object detected
in front
    {
      Robotstate = T90;
    }
    if (!LProx && !RProx && !CProx)   // No object detected left or right or in front
    {
      Robotstate = MFO;
    }
    if (LProx && RProx)   // Object detected left and right
    {
      Robotstate = ESC;
    }
    break;

    case MFO:
    val = analogRead(A7);
    bias = map(val, 0, 1023, -100, 100);
    // Serial.println(bias);
    Lspeed(Avspeed - bias);
    Rspeed(Avspeed + bias);
    delay(100);   // Allows for LProx and RProx to become true simultaneously
    Cservo.writeMicroseconds(1430);
    Cdistance = distanceSensor.measureDistanceCm();
    LProx = !(digitalRead(3));   // LProx is true when an object is detected
    RProx = !(digitalRead(2));   // RProx is true when an object is detected
    CProx = (Cdistance < Climit);   // CProx is true when an object is detected
    if (!LProx && !RProx && !CProx)   // No object detected left or right or in front
    {
      Robotstate = MFO;
    }
    if (LProx || RProx || CProx)   // Object detected left or right or in front
    {
      Robotstate = SAM;
    }
    break;

    case T90:
    Lspeed(0);
    Rspeed(0);
    Cservo.writeMicroseconds(1430);
    delay(500);
    Cservo.writeMicroseconds(2110);
    delay(500);
    Ldistance = distanceSensor.measureDistanceCm();
    Cservo.writeMicroseconds(790);
    delay(500);
    Rdistance = distanceSensor.measureDistanceCm();
    if (Ldistance >= Rdistance)   // turn left
    {
      Lspeed(-200);
      Rspeed(200);
      delay(600);
    }
    if (Ldistance < Rdistance)   // turn right
    {
```

```
      Lspeed(200);
      Rspeed(-200);
      delay(600);
    }
    Robotstate = SAM;
    break;

    case T15:
    Lspeed(0);
    Rspeed(0);
    if (LProx)   // turn right
    {
      Lspeed(200);
      Rspeed(-200);
      delay(100);
    }
    if (RProx)   // turn left
    {
      Lspeed(-200);
      Rspeed(200);
      delay(100);
    }
    Robotstate = SAM;
    break;

    case ESC:
    Lspeed(0);
    Rspeed(0);
    Lspeed(-200);
    Rspeed(-200);
    delay(1200);
    Lspeed(0);
    Rspeed(0);
    Lspeed(200);
    Rspeed(-200);
    delay(1200);
    Lspeed(0);
    Rspeed(0);
    Robotstate = SAM;
    break;

  }
}
```

Appendix E: Light seeking with two LDR sensors

```
/*
Nanolightseeker
Adrie Huesman
26 January 2022

Hardware
Arduino Nano Atmega328
Arduino Nano sensor I/O shield v3.0
4 Alkaline batteries so 6 volts
2 SG90 continous or 360 degrees servos
Wheel for Standard Servo 25T-40×7mm 2-Pack Pololu 4905
Ball Caster with 3/4" Plastic Ball Pololu 954
QTR-3A Reflectance Sensor Array Pololu 2456
Powerboard, own design based on 5V Step-Up/Step-Down Voltage Regulator S7V7F5 Pololu 2119
SG90 270 degrees servo and ultrasonic distance sensor HC-SR04
Chassis, own design 4mm aviation plywood 13 by 8 cm (two levels)

Software

I/O
D0 -- Reserved UART Rx
D1 -- Reserved UART Tx
D2 --
D3 --
D4 --
D5 Continuous servo left
D6 Continuous servo right
D7 --
D8 --
D9 270 degrees servo
D10 --
D11 --
D12 HC-SR04 Trig
D13 HC-SR04 Echo
A0 Sensor array left
A1 Sensor array center
A2 Sensor array right
A3 --
A4 -- Reserved I2C SDA
A5 -- Reserved I2C SDL
A6 LDR sensor left
A7 LDR sensor right

Other
SensorValues when all black 900 - 924, say 900
SensorValues when all white 488 - 579, say 600

Same RPMs achieved when:
Forward Lspeed 200 Rspeed 240
Backward Lspeed -250 Rspeed -200
Left turn Lspeed -200 Rspeed 200
Right turn Lspeed 200 Rspeed -200
 */

// Declare variables
int sensorValueL;     // variable to store the value coming from the left sensor
int sensorValueC;     // variable to store the value coming from the center sensor
int sensorValueR;     // variable to store the value coming from the right sensor
int Avspeed = 150;    // Average speed (200)
int LDRL;
int LDRR;
int Error;
float Kc = 1.0;      // Controller gain (0.40)
float Paction;

#include <Servo.h>
Servo Lservo;
Servo Rservo;
```

```
// Set speed of left motor, -500 < speed < 500
void Lspeed(int speed)
{
  Lservo.writeMicroseconds(speed + 1500);
}

// Set speed of right motor, -500 < speed < 500
void Rspeed(int speed)
{
  Rservo.writeMicroseconds(1500 - speed);
}

void setup()
{
  Serial.begin(9600);   // set up Serial library at 9600 bps
  Lservo.attach(5);
  Rservo.attach(6);
}

void loop()
{
  // Lspeed(200);
  // Rspeed(240);
  LDRL = analogRead(A6);
  LDRR = analogRead(A7) + 25;
  Serial.print(LDRL);
  Serial.print(',');
  Serial.print(LDRR);
  Serial.println();
  Error = LDRL - LDRR;
  Paction = Kc*Error;
  Lspeed(Avspeed + int(Paction));
  Rspeed(Avspeed - int(Paction) + 40);  // Motors behave different
  delay(100);
}
```

Appendix F: Bluetooth RC (proportional control)

```
/*
Adrie Huesman
HC05RobotB
To be used with Arduino Bluetooth Controlled Joystick, see
https://play.google.com/store/apps/details?id=uncia.robotics.joystick&hl=en_US&gl=US
Joystick (Advanced) mode
*/

// Declare variables
int Mspeed = 200;    // Motor speed
int hoek;
int sterkte;
int x;
int y;

// #include <SoftwareSerial.h>
// SoftwareSerial btSerial(9, 10); // RX, TX PIN

#include <Servo.h>
Servo Lservo;
Servo Rservo;

// Set speed of left motor, -500 < speed < 500
void Lspeed(int speed)
{
  Lservo.writeMicroseconds(speed + 1500);
}

// Set speed of right motor, -500 < speed < 500
void Rspeed(int speed)
{
  Rservo.writeMicroseconds(1500 - speed);
}

void setup() {
  pinMode(13,OUTPUT);
  Serial.begin(9600);
  // btSerial.begin(9600);
  Lservo.attach(5);
  Rservo.attach(6);
  // btSerial.println("Einde setup");
}

void loop()
{
  if(Serial.available() > 0) {
    String value = Serial.readStringUntil('#');
    if(value.length()==7)
    {
      String angle = value.substring(0, 3);
      String strength = value.substring(3, 6);
      String button = value.substring(6, 8);
      // Serial.print("angle: ");Serial.print(angle);Serial.print('\t');
      // Serial.print("strength: ");Serial.print(strength);Serial.print('\t');
      // Serial.print("button: ");Serial.print(button);Serial.println("");
      hoek = angle.toInt();
      sterkte = strength.toInt();
      // Serial.println(hoek);
      if (sterkte <= 10)
      {
        Lspeed(0);
        Rspeed(0);
      }
      if (sterkte > 10)
      {
        if ((hoek >= 0) && (hoek < 90))
        {
          Lspeed(Mspeed);
          x = hoek;
```

```
          y = map(x, 0, 90, -Mspeed, Mspeed);
          Rspeed(y);
        }
        if ((hoek >= 90) && (hoek < 180))
        {
          x = hoek;
          y = map(x, 90, 180, Mspeed, -Mspeed);
          Lspeed(y);
          Rspeed(Mspeed);
        }
        if ((hoek >= 180) && (hoek < 270))
        {
          Lspeed(-Mspeed);
          x = hoek;
          y = map(x, 180, 270, Mspeed, -Mspeed);
          Rspeed(y);
        }
        if ((hoek >= 270) && (hoek <= 360))
        {
          x = hoek;
          y = map(x, 270, 360, -Mspeed, Mspeed);
          Lspeed(y);
          Rspeed(-Mspeed);
        }
        Serial.flush();
        value="";
        delay(50);
      }
    }
  }
}
```