# include <Servo.h> // include the library of servo motor control

// define the control pin of each motor

const byte left\_front = 46;

const byte left\_rear = 47;

const byte right\_rear = 50;

const byte right\_front = 51;

// three machine states

enum STATE {

INITIALISING,

RUNNING,

STOPPED

};

// define motions states

enum MOTION{

FORWARD,

BACKWARD,

LEFT\_TURN,

RIGHT\_TURN,

LEFT\_ARC,

RIGHT\_ARC,

BACKWARD\_LEFT\_TURN,

};

// declare function output and function flag

MOTION cruise\_command;

int cruise\_output\_flag;

MOTION follow\_command;

int follow\_output\_flag;

MOTION avoid\_command;

int avoid\_output\_flag;

MOTION escape\_command;

int escape\_output\_flag;

MOTION motor\_input;

// define threshold of phototransistor difference

int photo\_dead\_zone = 5;

// define the sensor reading results

int photo\_left ;

int photo\_right;

int ir\_detect;

int bumper\_left;

int bumper\_right;

int bumper\_back;

// create servo objects for each motor

Servo left\_front\_motor;

Servo left\_rear\_motor;

Servo right\_rear\_motor;

Servo right\_front\_motor;

int speed\_val = 100;

int speed\_change;

void setup() {

Serial.begin(9600); // start serial communication

}

void loop() {

// put your main code here, to run repeatedly:

static STATE machine\_state = INITIALISING; // start from the sate INITIALIING

switch (machine\_state)

{

case INITIALISING:

machine\_state = initialising();

break;

case RUNNING:

machine\_state = running();

break;

case STOPPED:

machine\_state = stopped();

break;

}

}

STATE initialising(){

enable\_motors(); // enable motors

Serial.println("INITIALISING"); // print the current stage

return RUNNING; // return to RUNING STATE DIRECTLY

}

STATE running(){

//read\_serial\_command(); // read command from serial communication

speed\_change\_smooth(); //function to speed up and slow down smoothly

// this is just for test functions to read simulative sensor reading from monitor

serial\_read\_conditions();

// four function

cruise();

follow();

avoid();

escape();

// select the output command based on the function priority

arbitrate();

photo\_left = 0;

photo\_right = 0;

ir\_detect = 0;

bumper\_left = 0;

bumper\_right = 0;

bumper\_back = 0;

return RUNNING; // return to RUNNING STATE again, it will run the RUNNING

} // STATE REPEATLY

STATE stopped(){

disable\_motors(); // disable the motors

}

void speed\_change\_smooth() // change speed, called in RUNING STATE

{

speed\_val += speed\_change; // speed value add on speed change

if(speed\_val > 500) // make sure speed change less than 1000

speed\_val = 500;

speed\_change = 0; //make speed change equals 0 after updating the speed value

}

// cruise function output command and flag

void cruise()

{

cruise\_command = FORWARD;

cruise\_output\_flag=1;

}

// follow function output command and flag

void follow()

{ int delta;

//int left\_photo, right\_photo, delta;

//left\_photo=analog(1);

// right\_photo=analog(0);

delta=photo\_right - photo\_left;

if (abs(delta)>photo\_dead\_zone)

{if (delta>0)

follow\_command=LEFT\_TURN;

else

follow\_command=RIGHT\_TURN;

follow\_output\_flag=1;

}

else

follow\_output\_flag=0;

}

// avoid function output command and flag

void avoid()

{int val;

val=ir\_detect;

//val=ir\_detect();

if (val==1)

{avoid\_output\_flag=1;

avoid\_command=BACKWARD;}

else if (val==2)

{avoid\_output\_flag=1;

avoid\_command=RIGHT\_ARC;}

else if (val==3)

{avoid\_output\_flag=1;

avoid\_command=LEFT\_ARC;}

else

{avoid\_output\_flag=0;}

}

//escape function output command and flag

void escape()

{

//bumper\_check();

if (bumper\_left && bumper\_right)

{escape\_output\_flag=1;

escape\_command=BACKWARD\_LEFT\_TURN;

}

else if (bumper\_left)

{escape\_output\_flag=1;

escape\_command=RIGHT\_TURN;

}

else if (bumper\_right)

{escape\_output\_flag=1;

escape\_command=LEFT\_TURN;

}

else if (bumper\_back)

{escape\_output\_flag=1;

escape\_command=LEFT\_TURN;

}

else

escape\_output\_flag=0;

}

// check flag and select command based on priority

void arbitrate ()

{

if (cruise\_output\_flag==1)

{motor\_input=cruise\_command;}

if (follow\_output\_flag==1)

{motor\_input=follow\_command;}

if (avoid\_output\_flag ==1)

{motor\_input=avoid\_command;}

if (escape\_output\_flag==1)

{motor\_input=escape\_command;}

robotMove();

}

// connect the selected commands to the robot motions

void robotMove()

{

switch(motor\_input)

{

case FORWARD:

forward ();

delay(1000);

break;

case BACKWARD:

reverse ();

delay(1000);

break;

case LEFT\_TURN:

ccw();

delay(1000);

break;

case RIGHT\_TURN:

cw();

delay(1000);

break;

case LEFT\_ARC:

strafe\_left();

delay(1000);

break;

case RIGHT\_ARC:

strafe\_right();

delay(1000);

break;

case BACKWARD\_LEFT\_TURN:

reverse\_ccw();

delay(1000);

break;

}

}

// read simulative sensor reading

void serial\_read\_conditions()

{

char in\_data[8];

if(Serial.available()>0)

{

for(int k = 0; k < 9; k++)

{

char c = Serial.read();

in\_data[k] = c;

delay(2);

Serial.flush();

}

int temp0 = in\_data[0]-'0';

int temp1 = in\_data[1]-'0';

int temp2 = in\_data[2]-'0';

int temp3 = in\_data[3]-'0';

int temp4 = in\_data[4]-'0';

int temp5 = in\_data[5]-'0';

int temp6 = in\_data[6]-'0';

int temp7 = in\_data[7]-'0';

photo\_left = 10\*temp0 + temp1;

photo\_right = 10\*temp2 + temp3;

ir\_detect = temp4;

bumper\_left = temp5;

bumper\_right = temp6;

bumper\_back = temp7;

}

}

void disable\_motors(){ // function disable all motors, called in STOPPED STATE

left\_front\_motor.detach();

left\_rear\_motor.detach();

right\_rear\_motor.detach();

right\_front\_motor.detach();

pinMode(left\_front,INPUT); // set pinMode for next step

pinMode(left\_rear,INPUT);

pinMode(right\_rear,INPUT);

pinMode(right\_front,INPUT);

}

void enable\_motors() { //enable all motors, was called in INITIALZING SATE

left\_front\_motor.attach(left\_front);

left\_rear\_motor.attach(left\_rear);

right\_rear\_motor.attach(right\_rear);

right\_front\_motor.attach(right\_front);

}

void stop(){ // stop motors

left\_front\_motor.writeMicroseconds(1500);

left\_rear\_motor.writeMicroseconds(1500);

right\_rear\_motor.writeMicroseconds(1500);

right\_front\_motor.writeMicroseconds(1500);

}

void forward(){ // moving forward

left\_front\_motor.writeMicroseconds(1500 + speed\_val);

left\_rear\_motor.writeMicroseconds(1500 + speed\_val);

right\_rear\_motor.writeMicroseconds(1500 - speed\_val);

right\_front\_motor.writeMicroseconds(1500 - speed\_val);

}

void reverse(){ // reverse

left\_front\_motor.writeMicroseconds(1500 - speed\_val);

left\_rear\_motor.writeMicroseconds(1500 - speed\_val);

right\_rear\_motor.writeMicroseconds(1500 + speed\_val);

right\_front\_motor.writeMicroseconds(1500 + speed\_val);

}

void strafe\_left(){ // straight left

left\_front\_motor.writeMicroseconds(1500 - speed\_val);

left\_rear\_motor.writeMicroseconds(1500 + speed\_val);

right\_rear\_motor.writeMicroseconds(1500 + speed\_val);

right\_front\_motor.writeMicroseconds(1500 - speed\_val);

}

void strafe\_right(){ //straight right

left\_front\_motor.writeMicroseconds(1500 + speed\_val);

left\_rear\_motor.writeMicroseconds(1500 - speed\_val);

right\_rear\_motor.writeMicroseconds(1500 - speed\_val);

right\_front\_motor.writeMicroseconds(1500 + speed\_val);

}

void cw(){ //clockwise

left\_front\_motor.writeMicroseconds(1500 + speed\_val);

left\_rear\_motor.writeMicroseconds(1500 + speed\_val);

right\_rear\_motor.writeMicroseconds(1500 + speed\_val);

right\_front\_motor.writeMicroseconds(1500 + speed\_val);

}

void ccw(){ //anticlockwise

left\_front\_motor.writeMicroseconds(1500 - speed\_val);

left\_rear\_motor.writeMicroseconds(1500 - speed\_val);

right\_rear\_motor.writeMicroseconds(1500 - speed\_val);

right\_front\_motor.writeMicroseconds(1500 - speed\_val);

}

void reverse\_ccw()

{

left\_front\_motor.writeMicroseconds(1500 - speed\_val);

left\_rear\_motor.writeMicroseconds(1500 - speed\_val);

right\_rear\_motor.writeMicroseconds(1500 + speed\_val);

right\_front\_motor.writeMicroseconds(1500 + speed\_val);

delay(500);

left\_front\_motor.writeMicroseconds(1500 - speed\_val);

left\_rear\_motor.writeMicroseconds(1500 - speed\_val);

right\_rear\_motor.writeMicroseconds(1500 - speed\_val);

right\_front\_motor.writeMicroseconds(1500 - speed\_val);

}