**Tutorials and Worked Samples**

**for MECHENG 706 Design Projects**

Contents

[Motors 2](#_Toc1233482)

[Infrared sensor 8](#_Toc1233483)

[Gyro sensor 9](#_Toc1233484)

[Phototransistor 11](#_Toc1233485)

[Robot’s base code 12](#_Toc1233486)

# Motors

Codes (Modified from logan’s based code)

*# 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*

*};*

*// 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*

*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 > 1000) // make sure speed change less than 1000*

*speed\_val = 1000;*

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

*}*

*void read\_serial\_command() // this function is called in RUNING STATE*

*{*

*if (Serial.available()){*

*char val = Serial.read(); // get the input from serial communication*

*Serial.print("Speed:"); // print the current speed*

*Serial.print(speed\_val);*

*Serial.print(" ms ");*

*switch(val){*

*case 'w':*

*case 'W':*

*forward (); // call function forward if the command is “w” or “W”*

*Serial.println("Forward");*

*break;*

*case 's':*

*case 'S':*

*reverse (); // call function reverse if the command is “s” or “S”*

*Serial.println("Backward");*

*break;*

*case 'q':*

*case 'Q':*

*strafe\_left (); // call function strafe\_left if the command is “q” or “Q”*

*Serial.println("Strafe Left");*

*break;*

*case 'e':*

*case 'E':*

*strafe\_right (); // call function strafe\_right if the command is “e” or “E”*

*Serial.println("Strafe Rightt");*

*break;*

*case 'a':*

*case 'A':*

*ccw (); // call function ccw if the command is “a” or “A”*

*Serial.println("ccw");*

*break;*

*case 'd':*

*case 'D’*

*cw (); // call function cw if the command is “d” or “D”*

*Serial.println("cw");*

*break;*

*case '-':*

*case '\_':*

*speed\_change = -100; // make the speed change equals 100*

*Serial.println("-100");*

*break;*

*case '=':*

*case '+':*

*speed\_change = 100; // make the speed change equals 100*

*Serial.println("+");*

*break;*

*case 'p':*

*case 'P':*

*//default:*

*stop ();*

*Serial.println("stop");*

*break;*

*}*

*}*

*}*

*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*

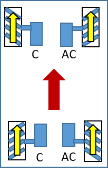
*left\_front\_motor.writeMicroseconds(1500);*

*left\_rear\_motor.writeMicroseconds(1500);*

*right\_rear\_motor.writeMicroseconds(1500);*

*right\_front\_motor.writeMicroseconds(1500);*

*}*

*void forward(){ // moving forward *

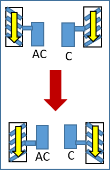
*left\_front\_motor.writeMicroseconds(1500 + speed\_val);*

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

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

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

*}*

*void reverse(){ // reverse *

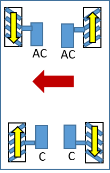
*left\_front\_motor.writeMicroseconds(1500 - speed\_val);*

*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 *

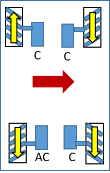
*left\_front\_motor.writeMicroseconds(1500 - speed\_val);*

*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 *

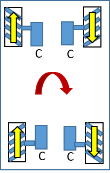
*left\_front\_motor.writeMicroseconds(1500 + speed\_val);*

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

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

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

*}*

*void cw(){ //clockwise *

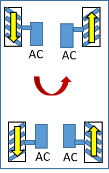
*left\_front\_motor.writeMicroseconds(1500 + speed\_val);*

*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);*

*}*

# Infrared sensor

*int irsensor = A0; //sensor is attached on pinA0*

*byte serialRead = 0; //for control serial communication*

*int signalADC = 0; // the read out signal in 0-1023 corresponding to 0-5v*

*void setup() {*

*// put your setup code here, to run once:*

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

*}*

*void loop() {*

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

*if (Serial.available()) // Check for input from terminal*

*{*

*serialRead = Serial.read(); // Read input*

*if (serialRead==49) // Check for flag to execute, 49 is ascii for 1, stop serial printing*

*{*

*Serial.end(); // end the serial communication to display sensor data on monitor*

*}*

*}*

*signalADC = analogRead(irsensor); // the read out is a signal from 0-1023 corresponding to 0-5v*

*int distance1 = 17948\*pow(signalADC,-1.22); // calculate the distance using the datasheet graph*

*int distancec = 46161\*pow(signalADC,-1.302); // calculate the distance using the calibrated graph*

*Serial.print("distance1 "); // print the results out using serial print*

*Serial.print(distance1);*

*Serial.println("cm ");*

*Serial.print("distancec ");*

*Serial.println(distancec);*

*Serial.println("cm");*

*}*

# Gyro sensor

*int sensorPin = A2; //define the pin that gyro is connected*

*int T = 100; // T is the time of one loop*

*int sensorValue = 0; // read out value of sensor*

*float gyroSupplyVoltage = 5; // supply voltage for gyro*

*float gyroZeroVoltage = 0; // the value of voltage when gyro is zero*

*float gyroSensitivity = 0.007; // gyro sensitivity unit is (mv/degree/second) get from datasheet*

*float rotationThreshold = 1.5; // because of gyro drifting, defining rotation angular velocity less than*

*// this value will not be ignored*

*float gyroRate = 0; // read out value of sensor in voltage*

*float currentAngle = 0; // current angle calculated by angular velocity integral on*

*byte serialRead = 0; // for serial print control*

*void setup() {*

*// put your setup code here, to run once:*

*Serial.begin(9600);*

*// this section is initialize the sensor, find the the value of voltage when gyro is zero*

*int i;*

*float sum = 0;*

*pinMode(sensorPin,INPUT);*

*Serial.println("please keep the sensor still for calibration");*

*Serial.println("get the gyro zero voltage");*

*for (i=0;i<100;i++) // read 100 values of voltage when gyro is at still, to calculate the zero-drift*

*{*

*sensorValue = analogRead(sensorPin);*

*sum += sensorValue;*

*delay(5);*

*}*

*gyroZeroVoltage = sum/100; // average the sum as the zero drifting*

*}*

*void loop() {*

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

*if (Serial.available()) // Check for input from terminal*

*{*

*serialRead = Serial.read(); // Read input*

*if (serialRead==49) // Check for flag to execute, 49 is asci for 1*

*{*

*Serial.end(); // end the serial communication to display the sensor data on monitor*

*}*

*}*

*// convert the 0-1023 signal to 0-5v*

*gyroRate = (analogRead(sensorPin)\*gyroSupplyVoltage)/1023;*

*// find the voltage offset the value of voltage when gyro is zero (still)\*

*gyroRate -= (gyroZeroVoltage/1023\*5);*

*// read out voltage divided the gyro sensitivity to calculate the angular velocity*

*float angularVelocity = gyroRate/ gyroSensitivity;*

*// if the angular velocity is less than the threshold, ignore it*

*if (angularVelocity >= rotationThreshold || angularVelocity <= -rotationThreshold)*

*{*

*// we are running a loop in T. one second will run (1000/T).*

*float angleChange = angularVelocity/(1000/T);*

*currentAngle += angleChange;*

*}*

*// keep the angle between 0-360*

*if (currentAngle < 0)*

*{currentAngle += 360;}*

*else if (currentAngle > 359)*

*{currentAngle -= 360;}*

*Serial.print(angularVelocity);*

*Serial.print(" ");*

*Serial.println(currentAngle);*

*// control the time per loop*

*delay (T);*

*}*

# Phototransistor

*int sensorValue = A1;*

*void setup() {*

*// put your setup code here, to run once:*

*Serial.begin(9600);*

*}*

*void loop() {*

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

*float volts = analogRead(sensorValue)\*5.0/1024.0;*

*Serial.print("analogy");*

*Serial.print(analogRead(sensorValue));*

*Serial.print(" ");*

*Serial.print("A1 = ");*

*Serial.println(volts);*

*delay(100);*

*}*

# Robot’s Base code

/\*

MechEng 706 Base Code

This code provides basic movement and sensor reading for the MechEng 706 Mecanum Wheel Robot Project

Hardware:

Arduino Mega2560 https://www.arduino.cc/en/Guide/ArduinoMega2560

MPU-9250 https://www.sparkfun.com/products/13762

Ultrasonic Sensor - HC-SR04 https://www.sparkfun.com/products/13959

Infrared Proximity Sensor - Sharp https://www.sparkfun.com/products/242

Infrared Proximity Sensor Short Range - Sharp https://www.sparkfun.com/products/12728

Servo - Generic (Sub-Micro Size) https://www.sparkfun.com/products/9065

Vex Motor Controller 29 https://www.vexrobotics.com/276-2193.html

Vex Motors https://www.vexrobotics.com/motors.html

Turnigy nano-tech 2200mah 2S https://hobbyking.com/en\_us/turnigy-nano-tech-2200mah-2s-25-50c-lipo-pack.html

Date: 11/11/2016

Author: Logan Stuart

Modified: 15/02/2018

Author: Logan Stuart

\*/

#include <Servo.h> //Need for Servo pulse output

#include <FaBo9Axis\_MPU9250.h> //refer to Installing9250Lib.png to install library

//#define NO\_READ\_MPU //Uncomment of MPU is not attached.

//#define NO\_HC-SR04 //Uncomment of HC-SR04 ultrasonic ranging sensor is not attached.

#define NO\_BATTERY\_V\_OK //Uncomment of BATTERY\_V\_OK if you do not care about battery damage.

//State machine states

enum STATE {

INITIALISING,

RUNNING,

STOPPED

};

//Refer to Shield Pinouts.jpg for pin locations

//Default motor control pins

const byte left\_front = 46;

const byte left\_rear = 47;

const byte right\_rear = 50;

const byte right\_front = 51;

//Default ultrasonic ranging sensor pins, these pins are defined my the Shield

const int TRIG\_PIN = 48;

const int ECHO\_PIN = 49;

// Anything over 400 cm (23200 us pulse) is "out of range". Hit:If you decrease to this the ranging sensor but the timeout is short, you may not need to read up to 4meters.

const unsigned int MAX\_DIST = 23200;

Servo left\_font\_motor; // create servo object to control Vex Motor Controller 29

Servo left\_rear\_motor; // create servo object to control Vex Motor Controller 29

Servo right\_rear\_motor; // create servo object to control Vex Motor Controller 29

Servo right\_font\_motor; // create servo object to control Vex Motor Controller 29

#ifndef NO\_READ\_NPU

FaBo9Axis fabo\_9axis;

bool axis\_OK;

#endif

int speed\_val = 100;

int speed\_change;

//Serial Pointer

HardwareSerial \*SerialCom;

void setup(void)

{

pinMode(LED\_BUILTIN, OUTPUT);

// The Trigger pin will tell the sensor to range find

pinMode(TRIG\_PIN, OUTPUT);

digitalWrite(TRIG\_PIN, LOW);

// Setup the Serial port and pointer, the pointer allows switching the debug info through the USB port(Serial) or Bluetooth port(Serial1) with ease.

SerialCom = &Serial1;

SerialCom->begin(115200);

SerialCom->println("MECHENG706\_Base\_Code\_25/01/2018");

delay(1000);

SerialCom->println("Setup....");

//If no MPU is attached then .begin will hang forever

#ifndef NO\_READ\_MPU

if (fabo\_9axis.begin()) {

SerialCom->println("configured FaBo 9Axis I2C Brick");

axis\_OK = true;

} else {

SerialCom->println("FaBo 9Axis device error");

axis\_OK = false;

}

#endif

delay(1000); //settling time but no really needed

}

void loop(void) //main loop

{

static STATE machine\_state = INITIALISING;

//Finite-state machine Code

switch (machine\_state) {

case INITIALISING:

machine\_state = initialising();

break;

case RUNNING: //Lipo Battery Volage OK

machine\_state = running();

break;

case STOPPED: //Stop of Lipo Battery voltage is too low, to protect Battery

machine\_state = stopped();

break;

};

}

STATE initialising() {

//initialising

SerialCom->println("INITIALISING....");

delay(1000); //One second delay to see the serial string "INITIALISING...."

SerialCom->println("Enabling Motors...");

enable\_motors();

SerialCom->println("RUNNING STATE...");

return RUNNING;

}

STATE running() {

static unsigned long previous\_millis;

read\_serial\_command();

fast\_flash\_double\_LED\_builtin();

if (millis() - previous\_millis > 500) { //Arduino style 500ms timed execution statement

previous\_millis = millis();

SerialCom->println("RUNNING---------");

speed\_change\_smooth();

Analog\_Range\_A4();

#ifndef NO\_READ\_MPU

if (axis\_OK)

MPU9250\_reading();

#endif

#ifndef NO\_HC-SR04

HC\_SR04\_range();

#endif

#ifndef NO\_BATTERY\_V\_OK

if (!is\_battery\_voltage\_OK()) return STOPPED;

#endif

}

return RUNNING;

}

//Stop of Lipo Battery voltage is too low, to protect Battery

STATE stopped() {

static byte counter\_lipo\_voltage\_ok;

static unsigned long previous\_millis;

disable\_motors();

slow\_flash\_LED\_builtin();

if (millis() - previous\_millis > 500) { //print massage every 500ms

previous\_millis = millis();

SerialCom->println("Lipo voltage too LOW, any lower and the lipo with be damaged");

SerialCom->println("Please Re-charge Lipo");

#ifndef NO\_BATTERY\_V\_OK

//500ms timed if statement to check lipo and output speed settings

if (is\_battery\_voltage\_OK()) {

SerialCom->print("Lipo OK Counter:");

SerialCom->println(counter\_lipo\_voltage\_ok);

counter\_lipo\_voltage\_ok++;

if (counter\_lipo\_voltage\_ok > 10) { //Making sure lipo voltage is stable

counter\_lipo\_voltage\_ok = 0;

enable\_motors();

SerialCom->println("Lipo OK returning to RUN STATE");

return RUNNING;

}

} else counter\_lipo\_voltage\_ok = 0;

#endif

}

return STOPPED;

}

void fast\_flash\_double\_LED\_builtin()

{

static byte indexer = 0;

static unsigned long fast\_flash\_millis;

if (millis() > fast\_flash\_millis) {

indexer++;

if (indexer > 4) {

fast\_flash\_millis = millis() + 700;

digitalWrite(LED\_BUILTIN, LOW);

indexer = 0;

} else {

fast\_flash\_millis = millis() + 100;

digitalWrite(LED\_BUILTIN, !digitalRead(LED\_BUILTIN));

}

}

}

void slow\_flash\_LED\_builtin()

{

static unsigned long slow\_flash\_millis;

if (millis() - slow\_flash\_millis > 2000) {

slow\_flash\_millis = millis();

digitalWrite(LED\_BUILTIN, !digitalRead(LED\_BUILTIN));

}

}

void speed\_change\_smooth()

{

speed\_val += speed\_change;

if (speed\_val > 1000)

speed\_val = 1000;

speed\_change = 0;

}

#ifndef NO\_BATTERY\_V\_OK

boolean is\_battery\_voltage\_OK()

{

static byte Low\_voltage\_counter;

static unsigned long previous\_millis;

int Lipo\_level\_cal;

int raw\_lipo;

//the voltage of a LiPo cell depends on its chemistry and varies from about 3.5V (discharged) = 717(3.5V Min) https://oscarliang.com/lipo-battery-guide/

//to about 4.20-4.25V (fully charged) = 860(4.2V Max)

//Lipo Cell voltage should never go below 3V, So 3.5V is a safety factor.

raw\_lipo = analogRead(A0);

Lipo\_level\_cal = (raw\_lipo - 717);

Lipo\_level\_cal = Lipo\_level\_cal \* 100;

Lipo\_level\_cal = Lipo\_level\_cal / 143;

if (Lipo\_level\_cal > 0 && Lipo\_level\_cal < 160) {

previous\_millis = millis();

SerialCom->print("Lipo level:");

SerialCom->print(Lipo\_level\_cal);

SerialCom->print("%");

SerialCom->print(" : Raw Lipo:");

SerialCom->println(raw\_lipo);

Low\_voltage\_counter = 0;

return true;

} else {

Low\_voltage\_counter++;

if (Low\_voltage\_counter > 5)

return false;

else

return true;

}

}

#endif

#ifndef NO\_HC-SR04

void HC\_SR04\_range()

{

unsigned long t1;

unsigned long t2;

unsigned long pulse\_width;

float cm;

float inches;

// Hold the trigger pin high for at least 10 us

digitalWrite(TRIG\_PIN, HIGH);

delayMicroseconds(10);

digitalWrite(TRIG\_PIN, LOW);

// Wait for pulse on echo pin

t1 = micros();

while ( digitalRead(ECHO\_PIN) == 0 ) {

t2 = micros();

pulse\_width = t2 - t1;

if ( pulse\_width > (MAX\_DIST + 1000)) {

SerialCom->println("HC-SR04: NOT found");

return;

}

}

// Measure how long the echo pin was held high (pulse width)

// Note: the micros() counter will overflow after ~70 min

t1 = micros();

while ( digitalRead(ECHO\_PIN) == 1)

{

t2 = micros();

pulse\_width = t2 - t1;

if ( pulse\_width > (MAX\_DIST + 1000) ) {

SerialCom->println("HC-SR04: Out of range");

return;

}

}

t2 = micros();

pulse\_width = t2 - t1;

// Calculate distance in centimeters and inches. The constants

// are found in the datasheet, and calculated from the assumed speed

//of sound in air at sea level (~340 m/s).

cm = pulse\_width / 58.0;

inches = pulse\_width / 148.0;

// Print out results

if ( pulse\_width > MAX\_DIST ) {

SerialCom->println("HC-SR04: Out of range");

} else {

SerialCom->print("HC-SR04:");

SerialCom->print(cm);

SerialCom->println("cm");

}

}

#endif

void Analog\_Range\_A4()

{

SerialCom->print("Analog Range A4:");

SerialCom->println(analogRead(A4));

}

#ifndef NO\_READ\_MPU

void MPU9250\_reading()

{

float ax, ay, az;

fabo\_9axis.readAccelXYZ(&ax, &ay, &az);

SerialCom->print("MPU ax:");

SerialCom->println(ax);

}

#endif

//Serial command pasing

void read\_serial\_command()

{

if (SerialCom->available()) {

char val = SerialCom->read();

SerialCom->print("Speed:");

SerialCom->print(speed\_val);

SerialCom->print(" ms ");

//Perform an action depending on the command

switch (val) {

case 'w'://Move Forward

case 'W':

forward ();

SerialCom->println("Forward");

break;

case 's'://Move Backwards

case 'S':

reverse ();

SerialCom->println("Backwards");

break;

case 'q'://Turn Left

case 'Q':

strafe\_left();

SerialCom->println("Strafe Left");

break;

case 'e'://Turn Right

case 'E':

strafe\_right();

SerialCom->println("Strafe Right");

break;

case 'a'://Turn Right

case 'A':

ccw();

SerialCom->println("ccw");

break;

case 'd'://Turn Right

case 'D':

cw();

SerialCom->println("cw");

break;

case '-'://Turn Right

case '\_':

speed\_change = -100;

SerialCom->println("-100");

break;

case '=':

case '+':

speed\_change = 100;

SerialCom->println("+");

break;

default:

stop();

SerialCom->println("stop");

break;

}

}

}

//----------------------Motor moments------------------------

//The Vex Motor Controller 29 use Servo Control signals to determine speed and direction, with 0 degrees meaning neutral https://en.wikipedia.org/wiki/Servo\_control

void disable\_motors()

{

left\_font\_motor.detach(); // detach the servo on pin left\_front to turn Vex Motor Controller 29 Off

left\_rear\_motor.detach(); // detach the servo on pin left\_rear to turn Vex Motor Controller 29 Off

right\_rear\_motor.detach(); // detach the servo on pin right\_rear to turn Vex Motor Controller 29 Off

right\_font\_motor.detach(); // detach the servo on pin right\_front to turn Vex Motor Controller 29 Off

pinMode(left\_front, INPUT);

pinMode(left\_rear, INPUT);

pinMode(right\_rear, INPUT);

pinMode(right\_front, INPUT);

}

void enable\_motors()

{

left\_font\_motor.attach(left\_front); // attaches the servo on pin left\_front to turn Vex Motor Controller 29 On

left\_rear\_motor.attach(left\_rear); // attaches the servo on pin left\_rear to turn Vex Motor Controller 29 On

right\_rear\_motor.attach(right\_rear); // attaches the servo on pin right\_rear to turn Vex Motor Controller 29 On

right\_font\_motor.attach(right\_front); // attaches the servo on pin right\_front to turn Vex Motor Controller 29 On

}

void stop() //Stop

{

left\_font\_motor.writeMicroseconds(1500);

left\_rear\_motor.writeMicroseconds(1500);

right\_rear\_motor.writeMicroseconds(1500);

right\_font\_motor.writeMicroseconds(1500);

}

void forward()

{

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

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

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

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

}

void reverse ()

{

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

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

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

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

}

void ccw ()

{

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

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

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

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

}

void cw ()

{

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

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

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

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

}

void strafe\_left ()

{

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

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

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

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

}

void strafe\_right ()

{

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

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

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

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

}