# Setup

The LCD shield used is the DFRobot LCD Keypad Shield v1.1.

The baud rate used is 9600 and the PrintMessage ends with “\r\n”.

# Code

#include <LiquidCrystal.h>

#include <avr/io.h>

#include <avr/interrupt.h>

#include <Math.h>

float newDistanceGoal;

boolean foundGoal = false;

int reachGoal = 0;

float distanceGoal;

float A,B,C;

float angleGoal;

LiquidCrystal lcd(8, 9, 4, 5, 6, 7);

double incomingByte;

volatile unsigned long int milliseconds = 0;

/\*\* Button inputs \*\*/

enum Buttons {

btnNONE,

btnRIGHT,

btnUP,

btnDOWN,

btnLEFT,

btnSELECT

};

Buttons whatbuttons; // Current button pressed

enum Mode {

mainMODE,

controlMODE,

sweepMODE,

wallMODE,

navMODE

};

Mode currentMode = mainMODE;

enum mainMenuState {

mainState,

controlState,

sweepState,

wallState,

navState

};

mainMenuState currentMenuState;

boolean startup = true;

ISR(TIMER2\_OVF\_vect) { //Chapter 16

//Register size = 64

// CLK = 62500 Hz

//Timer pertick = 1/CLK = 0.016ms

// from 0 to 64 = 64 \* 0.016ms = 1ms

milliseconds += 1; //increment every ms

}

void setup() {

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

timer\_Init();

sei(); //enable global interrupt

lcd.begin(16, 2);

lcd.setCursor(0, 0);

lcd.print("12878930");

lcd.setCursor(0,1);

lcd.print("Main menu");

//serial\_Init();

Serial.begin(9600);

Serial.setTimeout(350); // how long serial readString takes before timeout

ADC\_Init();

PrintMessage("CMD\_START"); // Start the "robot"

}

void loop() {

switch(currentMode) {

case mainMODE:

mainModeOperation();

break;

case controlMODE:

controlModeOperation();

break;

case sweepMODE:

sweepModeOperation();

break;

case wallMODE:

wallModeOperation();

break;

case navMODE:

navModeOperation();

break;

}

whatbuttons = readLCDButtons();

}

void PrintMessage(String message)

{

Serial.print(message);

Serial.write(13); //carriage return character (ASCII 13, or '\r')

Serial.write(10); //newline character (ASCII 10, or '\n')

mydelay(40);

}

void timer\_Init() {

//Timer2 register A = normal operation

TCCR2A &=~(1<<WGM20);

TCCR2A &=~(1<<WGM21);

TCCR2A &=~(1<<COM2B0);

TCCR2A &=~(1<<COM2B1);

TCCR2A &=~(1<<COM2A0);

TCCR2A &=~(1<<COM2A1);

// Prescaler for 64

TCCR2B &= ~(1<<CS20);

TCCR2B &= ~(1<<CS21);

TCCR2B |= (1<<CS22);

TIMSK2 |= (1<<TOIE2); //Enable Overflow interrupt

TIFR2 |= (1<<TOV2);

}

volatile unsigned long int mymillis() {

return milliseconds;

}

void mydelay(volatile long unsigned int delayTime) {

volatile long unsigned int count = mymillis();

while(mymillis() <= (delayTime + count)) {

}

}

Buttons readLCDButtons() {

static int inputButton;

inputButton = myAnalogRead(0);

mydelay(175); //DEBOUNCE

// read analog 0 with registers

if(inputButton > 1000) {

return btnNONE;

}

if(inputButton < 50) {

return btnRIGHT;

}

if(inputButton < 250) {

return btnUP;

}

if(inputButton < 450) {

return btnDOWN;

}

if(inputButton < 650) {

return btnLEFT;

}

if(inputButton < 850) {

return btnSELECT;

}

return btnNONE; // when all others fail, return this

}

void ADC\_Init() {

ADCSRA |= (1<<ADEN); // Enable ADC

ADMUX |= (1<<REFS0); // Internal Vcc 5v

}

int myAnalogRead(int Pin)

{

if(Pin == 1) {

ADMUX |= Pin; //Multiplexer for which pin to read from

}

else if(Pin == 0) {

ADMUX = 0;

ADMUX |= (1<<REFS0); // Internal Vcc 5v

}

ADCSRA |= (1<<ADSC); // start conversion

// wait for conversion to complete

while (!(ADCSRA &(1<<ADIF))); // becomes while(0) when the conversion is complete

ADCSRA |= (1<<ADIF);

return ADC;

}

void mainModeOperation() {

if(startup) {

lcd.clear();

lcd.setCursor(0,0);

lcd.print("12878930");

lcd.setCursor(0,1);

stateToText();

startup = false;

}

switch(whatbuttons) {

case btnDOWN:

if(currentMenuState == navState) {

currentMenuState = mainState;

}

else {

currentMenuState = currentMenuState + 1;

}

startup = true;

break;

case btnSELECT:

stateToMode();

startup = true;

break;

}

}

void stateToMode() { //State of the main menu (which mode is going to be selected)

switch(currentMenuState) {

case mainState:

break;

case controlState:

currentMode = controlMODE;

break;

case sweepState:

currentMode = sweepMODE;

break;

case wallState:

currentMode = wallMODE;

break;

case navState:

currentMode = navMODE;

break;

}

}

void stateToText() { //Printing for the main menu

switch(currentMenuState) {

case mainState:

lcd.print("Main menu");

break;

case controlState:

lcd.print("Control");

break;

case sweepState:

lcd.print("Sweep");

break;

case wallState:

lcd.print("Wall follow");

break;

case navState:

lcd.print("Navigation");

break;

}

}

void controlModeOperation() {

mydelay(120); // Adds a delay so that the robot responds the best to user input

switch(whatbuttons) {

case btnLEFT:

PrintMessage("CMD\_ACT\_ROT\_0\_10");

break;

case btnRIGHT:

PrintMessage("CMD\_ACT\_ROT\_1\_10");

break;

case btnUP:

PrintMessage("CMD\_ACT\_LAT\_1\_0.5\r\n");

break;

case btnDOWN:

PrintMessage("CMD\_ACT\_LAT\_0\_0.5\r\n");

break;

case btnSELECT:

currentMode = mainMODE;

currentMenuState = mainState;

startup = true;

break;

}

}

void sweepModeOperation() {

static boolean finished = false;

int minimumAngle = 0;

float minimum;

String currentString;

static float currentValue = 0;

float irValue;

int i = 0;

int cutString;

if(!finished) {

if(whatbuttons == btnUP) { // only starts when the btnUP is pressed

PrintMessage("CMD\_SEN\_ROT\_0");

for(i = 1; i <= 72;i++ ) { // scan the surrounding every 5 degrees (360/5 = 72)

PrintMessage("CMD\_SEN\_ROT\_" + (String) (360 - (i\*5))); // clockwise

mydelay(40);

PrintMessage("CMD\_SEN\_IR");

mydelay(30);

currentString = Serial.readString();

cutString = currentString.length();

currentString.remove(cutString-2); // remove the last 2 characters, for some reason made my program crash

irValue = currentString.toFloat();

mydelay(100);

if(irValue != irValue) { // if NaN

irValue = 5; // more than sensor range

}

if(((irValue <= minimum) ||(i == 1)) && (irValue != 0)) { //check if irValue isn't 0 as it does sometimes and is an outlier

minimumAngle = i\*5;

minimum = irValue;

}

if(irValue ==0) {

i--; // if the irValue equals 0, proceed to take the reading again

}

}

finished = true;

minimumAngle = 359 - minimumAngle;

for(int j = 0; j <= minimumAngle; j++) {

PrintMessage("CMD\_ACT\_ROT\_0\_1"); // move degrees by degrees to the shortest distance CCW

}

PrintMessage("CMD\_SEN\_ROT\_0");

}

else if(whatbuttons == btnSELECT) {

startup = true;

finished = false;

currentMode = mainMODE;

currentMenuState = mainState;

}

}

else {

if(whatbuttons == btnSELECT) {

startup = true;

finished = false;

currentMode = mainMODE;

currentMenuState = mainState;

}

}

}

void wallModeOperation() {

static boolean finished = false;

static int minimumAngle = 0;

static float minimum;

String currentString;

static float currentValue = 0;

static float irValue;

static float d1, d2, d1d2, dwall, dfront;

static float alpha = 15;

static int angle = 1;

int cutString;

static float error, oldError;

static float anglePosition = 0;

static boolean stopped = false;

static boolean firstCal = true;

static boolean badReading = false;

if(!stopped) {

if(!finished) { // while looking for the wall, does a sweep with 5 degrees precision

PrintMessage("CMD\_SEN\_ROT\_" + (String) (360 - (angle\*5)));

PrintMessage("CMD\_SEN\_IR");

currentString = Serial.readString();

mydelay(75);

cutString = currentString.length();

currentString.remove(cutString-2);

irValue = currentString.toFloat();

if(irValue != irValue) { // if NaN

irValue = 5; // more than sensor range

badReading = false;

}

else if(irValue == 0) { // if badReading

badReading = true;

}

else {

badReading = false;

}

if((irValue <= minimum) ||(angle == 1)) {

minimumAngle = angle\*5;

minimum = irValue;

}

if(angle == 72) {

minimumAngle = 360 - minimumAngle;

for(int j = 0; j <= (minimumAngle) ; j++) {

PrintMessage("CMD\_ACT\_ROT\_0\_1");

mydelay(20);

}

mydelay(75);

PrintMessage("CMD\_ACT\_ROT\_1\_90"); // turn 90 degrees to the wall (parallel)

anglePosition = 0; // 0 means that the program deemed the robot to be facing parallel to the nearest wall

finished = true;

}

if(!badReading) { // only increment the angle when the reading is deemed to be acceptable

angle++;

}

}

else {

PrintMessage("CMD\_SEN\_ROT\_90"); // check 90 degrees distance

mydelay(20);

PrintMessage("CMD\_SEN\_IR");

mydelay(20);

currentString = Serial.readString();

whatbuttons = readLCDButtons(); // adding readButtons regularly to exit when needed, bad practice but works

switch(whatbuttons) {

case btnUP:

firstCal = true;

stopped = true;

minimumAngle = minimum = anglePosition = angle = 0;

break;

case btnSELECT:

firstCal = true;

stopped = false;

PrintMessage("CMD\_SEN\_ROT\_0");

finished = false;

minimumAngle = minimum = anglePosition = angle = 0;

currentMode = mainMODE;

currentMenuState = mainState;

startup = true;

return;

}

cutString = currentString.length();

currentString.remove(cutString-2);

d1 = currentString.toFloat();

PrintMessage("CMD\_SEN\_ROT\_75"); // check the second measurement with alpha of 15 degrees

mydelay(20);

PrintMessage("CMD\_SEN\_IR");

mydelay(20);

currentString = Serial.readString();

whatbuttons = readLCDButtons();

switch(whatbuttons) {

case btnUP:

firstCal = true;

stopped = true;

minimumAngle = minimum = anglePosition = angle = 0;

break;

case btnSELECT:

firstCal = true;

stopped = false;

PrintMessage("CMD\_SEN\_ROT\_0");

finished = false;

minimumAngle = minimum = anglePosition = angle = 0;

currentMode = mainMODE;

currentMenuState = mainState;

startup = true;

return;

}

cutString = currentString.length();

currentString.remove(cutString-2);

d2 = currentString.toFloat();

if((d1 == d1) && (d2 == d2) && ((d1 != 0) || (d2 != 0))) { // making sure the 2 distances aren't obsolete or NaN

d1d2 = sqrt(d1\*d1+ d2\*d2 - 2\*d1\*d2\*cos((alpha\*3.14)/180));

//angleError = (asin(d1\*sin(30\*3.14/180)/d1d2))\*180/3.14;

dwall = d2\*d1\*sin((alpha\*3.14)/180)/d1d2;

//using equations from the lecture, find the real distance to the wall as it will rarely but purely parallel due to noise

whatbuttons = readLCDButtons();

switch(whatbuttons) {

case btnUP:

firstCal = true;

stopped = true;

minimumAngle = minimum = anglePosition = angle = 0;

break;

case btnSELECT:

firstCal = true;

stopped = false;

PrintMessage("CMD\_SEN\_ROT\_0");

finished = false;

minimumAngle = minimum = anglePosition = angle = 0;

currentMode = mainMODE;

currentMenuState = mainState;

startup = true;

return;

}

if(dwall != 0) { // if the distance is calculated properly

if(dwall > 2) {

error = dwall - 2; // using P of PID

if((error <= 0.15)) { // if the error is minimal

if(anglePosition < -8 && (error <= oldError)) { // avoids overshoot where the angle of the robot is too high

PrintMessage("CMD\_ACT\_ROT\_1\_" + (String) (error\*14));

anglePosition = anglePosition + error\*14;

}

else if((anglePosition > 0) || (oldError <= error)) { //if the angle position is a bit off and facing the wrong way

PrintMessage("CMD\_ACT\_ROT\_0\_" + (String) (error\*10));

anglePosition = anglePosition - error\*10;

}

}

else if(anglePosition > -14) { //limits the angle of turning to avoid hitting the wall

PrintMessage("CMD\_ACT\_ROT\_0\_" + (String) (error\*8));

anglePosition = anglePosition - error\*8;

}

else if(((error - oldError) < 0.05) && ((anglePosition < -10) || (anglePosition > 10))) {

anglePosition = 0; // if the anglePosition is really off, it will show after a bit and will be corrected by this statemetn

}

}

else { // if the robot is closer to the wall by 2 meters, do the same thing as when it is away but adapting it

error = 2 - dwall;

if (error <= 0.15){

if(anglePosition > 8 && (error <= oldError)) {

PrintMessage("CMD\_ACT\_ROT\_0\_" + (String) (error\*14));

anglePosition = anglePosition - error\*14;

}

else if(anglePosition < 0 || (oldError <= error)) {

PrintMessage("CMD\_ACT\_ROT\_1\_" + (String) (error\*10));

anglePosition = anglePosition + error\*10;

}

}

else if(anglePosition < 14) {

PrintMessage("CMD\_ACT\_ROT\_1\_" + (String) (error\*8));

anglePosition = anglePosition + error\*8;

}

else if(((error - oldError) < 0.05) && ((anglePosition < -10) || (anglePosition > 10))) {

anglePosition = 0;

}

}

}

// lcd.clear();

// lcd.setCursor(0,0);

// lcd.print(d1);

// lcd.print(" ");

// lcd.print(d2);

// lcd.print(" ");

// lcd.print(anglePosition);

// lcd.setCursor(0,1);

// lcd.print(dwall);

PrintMessage("CMD\_SEN\_ROT\_0");

PrintMessage("CMD\_SEN\_IR");

whatbuttons = readLCDButtons();

switch(whatbuttons) {

case btnUP:

firstCal = true;

stopped = true;

minimumAngle = minimum = anglePosition = angle = 0;

break;

case btnSELECT:

firstCal = true;

stopped = false;

PrintMessage("CMD\_SEN\_ROT\_0");

finished = false;

minimumAngle = minimum = anglePosition = angle = 0;

currentMode = mainMODE;

currentMenuState = mainState;

startup = true;

return;

}

currentString = Serial.readString();

cutString = currentString.length();

currentString.remove(cutString-2);

dfront = currentString.toFloat();

if((dfront == dfront) && (dfront != 0)) { // if there's a wall ahead

if(dfront <= 2.4) {

PrintMessage("CMD\_ACT\_ROT\_1\_90"); // turn 90 degrees

}

}

PrintMessage("CMD\_ACT\_LAT\_1\_0.5");

oldError = error; // new error is recorded

}

}

}

switch(whatbuttons) {

case btnUP:

firstCal = true;

stopped = true;

minimumAngle = minimum = anglePosition = angle = 0;

break;

case btnSELECT:

firstCal = true;

stopped = false;

PrintMessage("CMD\_SEN\_ROT\_0");

finished = false;

minimumAngle = minimum = anglePosition = angle = 0;

currentMode = mainMODE;

currentMenuState = mainState;

startup = true;

return;

}

}

void navModeOperation() {

float leftCorner, rightCorner;

float leftAngle, rightAngle;

float oldLeft, oldRight;

float maxLeft, maxRight;

float leftDistance, rightDistance;

float frontDistance;

String currentString;

int cutString;

int collisions;

boolean wallWarning = false;

float tempGoal;

if(!foundGoal) {

do {

PrintMessage("CMD\_SEN\_PING");

mydelay(100);

currentString = Serial.readString();

cutString = currentString.length();

currentString.remove(cutString-2);

distanceGoal = currentString.toFloat();

PrintMessage("CMD\_SEN\_PING");

mydelay(100);

currentString = Serial.readString();

cutString = currentString.length();

currentString.remove(cutString-2);

tempGoal = currentString.toFloat();

} while(distanceGoal != tempGoal);

if(distanceGoal != 0) { // will return 0 if it outside of the scope

if(distanceGoal < 4.4) { // if within 4.4, it found the goal + some error margin

foundGoal = true;

}

}

whatbuttons = readLCDButtons();

if(whatbuttons == btnSELECT) {

startup = true;

foundGoal = false;

reachGoal = 0;

distanceGoal = 0;

A = B = C = angleGoal =0;

newDistanceGoal = 0;

currentMode = mainMODE;

currentMenuState = mainState;

return;

}

PrintMessage("CMD\_SEN\_ROT\_0");

mydelay(30);

PrintMessage("CMD\_SEN\_IR");

mydelay(50);

currentString = Serial.readString();

cutString = currentString.length();

currentString.remove(cutString-2);

frontDistance = currentString.toFloat(); // check front distance

if(frontDistance != frontDistance) { // if the frontdistance is less than 5m

PrintMessage("CMD\_SEN\_ROT\_20"); // check leftcorner distance to avoid clipping corners

PrintMessage("CMD\_SEN\_IR");

currentString = Serial.readString();

cutString = currentString.length();

currentString.remove(cutString-2);

leftCorner = currentString.toFloat();

whatbuttons = readLCDButtons();

if(whatbuttons == btnSELECT) {

startup = true;

foundGoal = false;

reachGoal = 0;

distanceGoal = 0;

A = B = C = angleGoal =0;

newDistanceGoal = 0;

currentMode = mainMODE;

currentMenuState = mainState;

return;

}

PrintMessage("CMD\_SEN\_ROT\_340");

PrintMessage("CMD\_SEN\_IR");

currentString = Serial.readString();

cutString = currentString.length();

currentString.remove(cutString-2);

rightCorner = currentString.toFloat(); // check right corner distance to avoid clipping corners

whatbuttons = readLCDButtons();

if(whatbuttons == btnSELECT) {

startup = true;

foundGoal = false;

reachGoal = 0;

distanceGoal = 0;

A = B = C = angleGoal =0;

newDistanceGoal = 0;

currentMode = mainMODE;

currentMenuState = mainState;

return;

}

if((rightCorner != rightCorner) && (leftCorner != leftCorner)) { // if both corners are safe

PrintMessage("CMD\_ACT\_LAT\_1\_2");

}

else if((rightCorner != rightCorner) && (leftCorner == leftCorner)) { // if right corner is more safe

if(leftCorner < 2) {

PrintMessage("CMD\_ACT\_ROT\_1\_10");// turn slightly to avoid clipping the wall

}

PrintMessage("CMD\_ACT\_LAT\_1\_1");

}

else if((rightCorner == rightCorner) && (leftCorner != leftCorner)) { // if left corner is safer

if(rightCorner < 2) {

PrintMessage("CMD\_ACT\_ROT\_0\_10");

}

PrintMessage("CMD\_ACT\_LAT\_1\_1");

}

else if((rightCorner > 0.75) && (leftCorner > 0.75)) { // if it is within good distance

PrintMessage("CMD\_ACT\_LAT\_1\_1");

}

else if((rightCorner > 0.75) && (leftCorner < 0.75)) { // if leftcorner unsafe turn 90 degrees to the right

PrintMessage("CMD\_ACT\_ROT\_1\_90");

}

else if((rightCorner < 0.75) && (leftCorner > 0.75)) { // if right corner unsafe turn left

PrintMessage("CMD\_ACT\_ROT\_0\_90");

}

whatbuttons = readLCDButtons();

if(whatbuttons == btnSELECT) {

startup = true;

foundGoal = false;

reachGoal = 0;

distanceGoal = 0;

A = B = C = angleGoal =0;

newDistanceGoal = 0;

currentMode = mainMODE;

currentMenuState = mainState;

return;

}

}

else {

if(frontDistance >= 2) { // if the front distance is bigger than 2

PrintMessage("CMD\_SEN\_ROT\_20");

PrintMessage("CMD\_SEN\_IR");

currentString = Serial.readString();

cutString = currentString.length();

currentString.remove(cutString-2);

leftCorner = currentString.toFloat(); // check leftcorner

if(whatbuttons == btnSELECT) {

startup = true;

foundGoal = false;

reachGoal = 0;

distanceGoal = 0;

A = B = C = angleGoal =0;

newDistanceGoal = 0;

currentMode = mainMODE;

currentMenuState = mainState;

return;

}

PrintMessage("CMD\_SEN\_ROT\_340"); // check right corner

PrintMessage("CMD\_SEN\_IR");

currentString = Serial.readString();

cutString = currentString.length();

currentString.remove(cutString-2);

rightCorner = currentString.toFloat();

whatbuttons = readLCDButtons();

if(whatbuttons == btnSELECT) {

startup = true;

foundGoal = false;

reachGoal = 0;

distanceGoal = 0;

A = B = C = angleGoal =0;

newDistanceGoal = 0;

currentMode = mainMODE;

currentMenuState = mainState;

return;

}

// check the corners values to change if needed bc of clipping corners

if((rightCorner != rightCorner) && (leftCorner != leftCorner)) {

PrintMessage("CMD\_ACT\_LAT\_1\_" + (String)(frontDistance - 1.25));

}

else if((rightCorner != rightCorner) && (leftCorner == leftCorner)) {

if(leftCorner <2) {

PrintMessage("CMD\_ACT\_ROT\_1\_10");

}

PrintMessage("CMD\_ACT\_LAT\_1\_" + (String)(frontDistance - 1.25));

}

else if((rightCorner == rightCorner) && (leftCorner != leftCorner)) {

if(rightCorner < 2) {

PrintMessage("CMD\_ACT\_ROT\_0\_10");

}

PrintMessage("CMD\_ACT\_LAT\_1\_" + (String)(frontDistance - 1.25));

}

else if((rightCorner > 0.75) && (leftCorner > 0.75)) {

PrintMessage("CMD\_ACT\_LAT\_1\_" + (String)(frontDistance - 1.25));

}

else if((rightCorner > 0.75) && (leftCorner < 0.75)) {

PrintMessage("CMD\_ACT\_ROT\_1\_30");

}

else if((rightCorner < 0.75) && (leftCorner > 0.75)) {

PrintMessage("CMD\_ACT\_ROT\_0\_30");

}

whatbuttons = readLCDButtons();

if(whatbuttons == btnSELECT) {

startup = true;

foundGoal = false;

reachGoal = 0;

distanceGoal = 0;

A = B = C = angleGoal =0;

newDistanceGoal = 0;

currentMode = mainMODE;

currentMenuState = mainState;

return;

}

}

else { // if front distance is smaller than 2

for(int i = 1; i < 9; i++) { //check left distance every 10 degrees

PrintMessage("CMD\_SEN\_ROT\_" + (String) (i\*10));

mydelay(30);

PrintMessage("CMD\_SEN\_IR");

mydelay(30);

currentString = Serial.readString();

cutString = currentString.length();

currentString.remove(cutString-2);

leftDistance = currentString.toFloat();

whatbuttons = readLCDButtons();

if(whatbuttons == btnSELECT) {

startup = true;

foundGoal = false;

reachGoal = 0;

distanceGoal = 0;

A = B = C = angleGoal =0;

newDistanceGoal = 0;

currentMode = mainMODE;

currentMenuState = mainState;

return;

}

if(leftDistance != leftDistance) { // if leftDistance > 5, end the loop

// PrintMessage("CMD\_ACT\_ROT\_0\_" + (String) (i\*10));

// PrintMessage("CMD\_ACT\_LAT\_1\_2");

leftAngle = i\*10;

i = 8;

}

else {

if(leftDistance > oldLeft) {

maxLeft = leftDistance;

leftAngle = i\*10;

}

oldLeft = leftDistance;

}

}

for(int j = 1; j < 9; j++) { // same logic for right distance

PrintMessage("CMD\_SEN\_ROT\_" + (String) (360-(j\*10)));

mydelay(30);

PrintMessage("CMD\_SEN\_IR");

mydelay(30);

currentString = Serial.readString();

cutString = currentString.length();

currentString.remove(cutString-2);

rightDistance = currentString.toFloat();

if(whatbuttons == btnSELECT) {

startup = true;

foundGoal = false;

reachGoal = 0;

distanceGoal = 0;

A = B = C = angleGoal =0;

newDistanceGoal = 0;

currentMode = mainMODE;

currentMenuState = mainState;

return;

}

if(rightDistance != rightDistance) {

// PrintMessage("CMD\_ACT\_ROT\_1\_" + (String) (j\*10));

// PrintMessage("CMD\_ACT\_LAT\_1\_2");

rightAngle = j\*10;

j = 8;

}

else {

if(rightDistance > oldRight) {

maxRight = rightDistance;

rightAngle = j\*10;

}

oldRight = rightDistance;

}

}

if(whatbuttons == btnSELECT) {

startup = true;

foundGoal = false;

reachGoal = 0;

distanceGoal = 0;

A = B = C = angleGoal =0;

newDistanceGoal = 0;

currentMode = mainMODE;

currentMenuState = mainState;

return;

}

if((rightDistance == rightDistance) && (leftDistance == leftDistance) && (leftDistance != 0) && (rightDistance != 0)) {

if((frontDistance > maxRight) && (frontDistance > maxLeft) && (frontDistance > 1)) { //check which distance is hgiher

PrintMessage("CMD\_ACT\_LAT\_1\_" + (String) (frontDistance -1));

mydelay(30);

}

if(maxRight > maxLeft) {

if(maxRight < 1.75) {

PrintMessage("CMD\_ACT\_ROT\_1\_180"); // if not safe turn around

mydelay(30);

}

else {

PrintMessage("CMD\_ACT\_ROT\_1\_" + (String) rightAngle);

mydelay(50);

PrintMessage("CMD\_ACT\_LAT\_1\_" + (String) (maxRight -1));

mydelay(50);

}

}

else {

if(maxLeft < 1.75) {

PrintMessage("CMD\_ACT\_ROT\_1\_180"); // if not safe turn around

mydelay(30);

}

else {

PrintMessage("CMD\_ACT\_ROT\_0\_" + (String) leftAngle);

mydelay(50);

PrintMessage("CMD\_ACT\_LAT\_1\_" + (String) (maxLeft -1));

mydelay(50);

}

}

}

else if((rightDistance != rightDistance) || (leftDistance != leftDistance)){

if(rightAngle > leftAngle) { // check angle so that the robot can go as straight as possible

PrintMessage("CMD\_ACT\_ROT\_0\_" + (String) (leftAngle));

}

else {

PrintMessage("CMD\_ACT\_ROT\_1\_" + (String) (rightAngle));

}

}

}

}

}

else { //if within 5 m

switch(reachGoal) {

case 0:

do { // do while to check if the goal ping is the right values and not errors, done later too

PrintMessage("CMD\_SEN\_PING");

mydelay(100);

currentString = Serial.readString();

cutString = currentString.length();

currentString.remove(cutString-2);

distanceGoal = currentString.toFloat();

PrintMessage("CMD\_SEN\_PING");

mydelay(100);

currentString = Serial.readString();

cutString = currentString.length();

currentString.remove(cutString-2);

tempGoal = currentString.toFloat();

} while(distanceGoal != tempGoal);

if(distanceGoal <= 0.8) { // if within 0.8, really close to the gal

reachGoal = 3;

}

A = distanceGoal;

PrintMessage("CMD\_SEN\_ROT\_0");

mydelay(30);

PrintMessage("CMD\_SEN\_IR");

mydelay(50);

currentString = Serial.readString();

cutString = currentString.length();

currentString.remove(cutString-2);

frontDistance = currentString.toFloat(); //check front distance

if((frontDistance != frontDistance) || (frontDistance > 1)) { //make sure it is safe

PrintMessage("CMD\_SEN\_ROT\_20");

mydelay(30);

PrintMessage("CMD\_SEN\_IR");

mydelay(30);

currentString = Serial.readString();

mydelay(75);

whatbuttons = readLCDButtons();

if(whatbuttons == btnSELECT) {

startup = true;

foundGoal = false;

reachGoal = 0;

distanceGoal = 0;

A = B = C = angleGoal =0;

newDistanceGoal = 0;

currentMode = mainMODE;

currentMenuState = mainState;

return;

}

cutString = currentString.length();

currentString.remove(cutString-2);

leftCorner = currentString.toFloat(); // check left corner

PrintMessage("CMD\_SEN\_ROT\_340");

mydelay(30);

PrintMessage("CMD\_SEN\_IR");

mydelay(30);

currentString = Serial.readString();

mydelay(75);

whatbuttons = readLCDButtons();

if(whatbuttons == btnSELECT) {

startup = true;

foundGoal = false;

reachGoal = 0;

distanceGoal = 0;

A = B = C = angleGoal =0;

newDistanceGoal = 0;

currentMode = mainMODE;

currentMenuState = mainState;

return;

}

cutString = currentString.length();

currentString.remove(cutString-2);

rightCorner = currentString.toFloat(); // check right corner

if((rightCorner != rightCorner) || (leftCorner != leftCorner)) {

//PrintMessage("CMD\_ACT\_LAT\_1\_0.5");

}

else if((rightCorner < 0.75) && (leftCorner < 0.75)) {

PrintMessage("CMD\_ACT\_ROT\_1\_180");

wallWarning = true;

}

mydelay(50);

do { // check the values of the ping twice to make sure that it is valid

PrintMessage("CMD\_SEN\_PING");

mydelay(100);

currentString = Serial.readString();

cutString = currentString.length();

currentString.remove(cutString-2);

distanceGoal = currentString.toFloat();

PrintMessage("CMD\_SEN\_PING");

mydelay(100);

currentString = Serial.readString();

cutString = currentString.length();

currentString.remove(cutString-2);

tempGoal = currentString.toFloat();

} while(distanceGoal != tempGoal);

if(distanceGoal <= 0.8) {

reachGoal = 3;

}

mydelay(100);

if(distanceGoal == 0) {

foundGoal = false; // if out of range, go bad to finding the goal (wandering)

}

else if(!wallWarning) { // if no wall next to it

whatbuttons = readLCDButtons();

if(whatbuttons == btnSELECT) {

startup = true;

foundGoal = false;

reachGoal = 0;

distanceGoal = 0;

A = B = C = angleGoal =0;

newDistanceGoal = 0;

currentMode = mainMODE;

currentMenuState = mainState;

return;

}

PrintMessage("CMD\_ACT\_LAT\_1\_0.5");

mydelay(30);

C = distanceGoal;

B = 0.52;

angleGoal = acos((B\*B+C\*C-A\*A)/(2\*B\*C)); // calculate the angle to turn towards the goal with trig

while(angleGoal > 1)

{

C += 0.1;

angleGoal = acos((B\*B+C\*C-A\*A)/(2\*B\*C)); // calculate the angle to turn towards the goal with trig

}

angleGoal = 180 - (angleGoal \*(180/3.14));

if((angleGoal != 0) && (angleGoal == angleGoal)) { // if good reading, turn

PrintMessage("CMD\_ACT\_ROT\_1\_" + (String) (angleGoal));

reachGoal++;

}

}

}

else {

PrintMessage("CMD\_ACT\_ROT\_0\_20"); // turn 20 degrees if there is a wall

mydelay(100);

}

break;

case 1: // after triangle, find whether the robot has to turn right or left to face the goal

PrintMessage("CMD\_SEN\_IR");

mydelay(50);

currentString = Serial.readString();

cutString = currentString.length();

currentString.remove(cutString-2);

frontDistance = currentString.toFloat();

mydelay(100);

if((frontDistance != frontDistance) || (frontDistance > 1)) {

PrintMessage("CMD\_ACT\_LAT\_1\_0.2");

do {

mydelay(50);

PrintMessage("CMD\_SEN\_PING");

mydelay(50);

currentString = Serial.readString();

cutString = currentString.length();

currentString.remove(cutString-2);

newDistanceGoal = currentString.toFloat();

mydelay(50);

PrintMessage("CMD\_SEN\_PING");

mydelay(50);

currentString = Serial.readString();

cutString = currentString.length();

currentString.remove(cutString-2);

tempGoal = currentString.toFloat();

} while(tempGoal != newDistanceGoal);

if(newDistanceGoal <= 0.8) {

reachGoal = 3 ;

}

if(distanceGoal <= 0.8) {

reachGoal = 3;

}

whatbuttons = readLCDButtons();

if(whatbuttons == btnSELECT) {

startup = true;

foundGoal = false;

reachGoal = 0;

distanceGoal = 0;

A = B = C = angleGoal =0;

newDistanceGoal = 0;

currentMode = mainMODE;

currentMenuState = mainState;

return;

}

mydelay(100);

// if new distance is bigger; wrong angle and go the other way

if((newDistanceGoal > distanceGoal) && (newDistanceGoal == newDistanceGoal) && (newDistanceGoal != 0)) {

mydelay(50);

PrintMessage("CMD\_ACT\_ROT\_0\_" + (String) (angleGoal\*2)); // wrong angle so turn the other way towards the goal

mydelay(50);

PrintMessage("CMD\_ACT\_LAT\_1\_0.2");

reachGoal++;

}

else if(newDistanceGoal < distanceGoal){ // right angle

PrintMessage("CMD\_ACT\_LAT\_0\_0.2");

reachGoal++;

}

}

else { // if theres a wall or bad reading

if(frontDistance != 0) {

PrintMessage("CMD\_ACT\_ROT\_0\_180");

}

else {

PrintMessage("CMD\_ACT\_ROT\_0\_45");

reachGoal = 0;

}

mydelay(100);

//reachGoal = 0;

}

break;

case 2: // now checks for the distance between goal and robot and move towards it 0.5 away

PrintMessage("CMD\_SEN\_IR");

mydelay(50);

currentString = Serial.readString();

cutString = currentString.length();

currentString.remove(cutString-2);

frontDistance = currentString.toFloat(); // check front distance

whatbuttons = readLCDButtons();

if(whatbuttons == btnSELECT) {

startup = true;

foundGoal = false;

reachGoal = 0;

distanceGoal = 0;

A = B = C = angleGoal =0;

newDistanceGoal = 0;

currentMode = mainMODE;

currentMenuState = mainState;

return;

}

mydelay(100);

// if front distance isn't an error and bigger than the distance to goal ( no walls)

if((frontDistance != frontDistance)||((frontDistance != 0) && ((frontDistance == frontDistance) && (frontDistance > newDistanceGoal)))) {

mydelay(100);

if(frontDistance > 0.8) {

PrintMessage("CMD\_ACT\_LAT\_1\_" + (String) (newDistanceGoal - 0.5)); // reach the goal

}

do { // check the ping twice for validity

PrintMessage("CMD\_SEN\_PING");

mydelay(100);

currentString = Serial.readString();

cutString = currentString.length();

currentString.remove(cutString-2);

distanceGoal = currentString.toFloat();

PrintMessage("CMD\_SEN\_PING");

mydelay(100);

currentString = Serial.readString();

cutString = currentString.length();

currentString.remove(cutString-2);

tempGoal = currentString.toFloat();

} while(distanceGoal != tempGoal);

if(distanceGoal <= 0.8) {

reachGoal++;

}

else {

reachGoal = 0;

}

}

else {

if((frontDistance != 0)) { // if front distance gives an error, start again

reachGoal = 0;

}

}

break;

case 3: // goal is found by the robot and print finished

lcd.clear();

lcd.setCursor(0,0);

lcd.print("Finished");

lcd.setCursor(0,1);

lcd.print("Navigation");

//reachGoal++;

break;

case 4:

lcd.clear();

lcd.setCursor(0,0);

lcd.print("Finished");

lcd.setCursor(0,1);

lcd.print("Navigation");

break;

}

}

if(whatbuttons == btnSELECT) {

startup = true;

foundGoal = false;

reachGoal = 0;

distanceGoal = 0;

A = B = C = angleGoal =0;

newDistanceGoal = 0;

currentMode = mainMODE;

currentMenuState = mainState;

return;

}

}