#include <FEHLCD.h>

#include <FEHIO.h>

#include <FEHUtility.h>

#include <FEHMotor.h>

#include <FEHRPS.h>

#include <FEHServo.h>

//Declarations for encoders & motors & servo

ButtonBoard buttons(FEHIO::Bank3);

AnalogInputPin right\_encoder(FEHIO::P0\_0);

AnalogInputPin left\_encoder(FEHIO::P0\_1);

FEHMotor right\_motor(FEHMotor::Motor0);

FEHMotor left\_motor(FEHMotor::Motor1);

FEHMotor scoop(FEHMotor::Motor2);//negative=retract

FEHMotor crank(FEHMotor::Motor3);//positive = clockwise

AnalogInputPin cds (FEHIO::P0\_2);

FEHServo servo (FEHServo::Servo0);

void counting(int counter)

{

int r = 0, l = 0, n = 0, m=0, starttime = TimeNow();

while(n < counter)

{

if(right\_encoder.Value()>1.75&&left\_encoder.Value()>1.75&&m==0)

{

r++;

m=1;

}

if(right\_encoder.Value()>1.75&&left\_encoder.Value()<1.75&&m==0)

{

l++;

m=2;

}

if(right\_encoder.Value()<1.75&&left\_encoder.Value()>1.75&&m==0)

{

r++;

m=3;

}

if(right\_encoder.Value()<1.75&&left\_encoder.Value()<1.75&&m==0)

{

l++;

m=4;

}

if(right\_encoder.Value()>1.75&&left\_encoder.Value()<1.75&&m==1)

{

l++;

m=2;

}

if(right\_encoder.Value()<1.75&&left\_encoder.Value()>1.75&&m==1)

{

r++;

m=3;

}

if(right\_encoder.Value()<1.75&&left\_encoder.Value()<1.75&&m==2)

{

r++;

m=4;

}

if(right\_encoder.Value()<1.75&&left\_encoder.Value()<1.75&&m==3)

{

l++;

m=4;

}

if(right\_encoder.Value()<1.75&&left\_encoder.Value()>1.75&&m==4)

{

l++;

m=3;

}

if(right\_encoder.Value()>1.75&&left\_encoder.Value()<1.75&&m==4)

{

r++;

m=2;

}

if(right\_encoder.Value()>1.75&&left\_encoder.Value()>1.75&&m==3)

{

r++;

m=1;

}

if(right\_encoder.Value()>1.75&&left\_encoder.Value()>1.75&&m==2)

{

l++;

m=1;

}

// {

// if (right\_encoder.Value() > 3.0)

// {

// r = r + 1;

// }

// else if(right\_encoder.Value() < 0.3)

// {

// r = r + 1;

// }

// }

// if (left\_encoder.Value() > 3.0)

// {

// l = l + 1;

// }

// else if(left\_encoder.Value() < 0.3)

// {

// l = l + 1;

// }

n = (r + l) / 2;

if (TimeNow() - starttime > 10000)

{

n=1000;

}

} //End while

}

void move\_forward(int percent, int counts) //using encoders

{

//Set both motors to desired percent

right\_motor.SetPercent(percent);

left\_motor.SetPercent(percent);

//While the average of the left and right encoder are less than counts,

//keep running motors

counting(counts);

//Turn off motors

right\_motor.Stop();

left\_motor.Stop();

}

void move\_forwardtime(int percent, int counts) //using encoders

{

//Set both motors to desired percent

right\_motor.SetPercent(percent);

left\_motor.SetPercent(percent);

//While the average of the left and right encoder are less than counts,

//keep running motors

counting(counts);

//Turn off motors

right\_motor.Stop();

left\_motor.Stop();

}

void move\_backward(int percent, int counts) //using encoders

{

//Set both motors to desired percent

right\_motor.SetPercent(-percent - 2);

left\_motor.SetPercent(-percent);

//While the average of the left and right encoder are less than counts,

//keep running motors

counting(counts);

//Turn off motors

right\_motor.Stop();

left\_motor.Stop();

}

void turn\_right(int percent, int counts) //using encoders

{

//Set both motors to desired percent

//hint: set right motor backwards, left motor forwards

right\_motor.SetPercent(-1\*percent);

left\_motor.SetPercent(percent);

//While the average of the left and right encoder are less than counts,

//keep running motors

counting(counts);

//Turn off motors

right\_motor.Stop();

left\_motor.Stop();

}

void turn\_left(int percent, int counts) //using encoders

{

//Set both motors to desired percent

right\_motor.SetPercent(percent);

left\_motor.SetPercent(-1\*percent);

//While the average of the left and right encoder are less than counts,

//keep running motors

counting(counts);

//Turn off motors

right\_motor.Stop();

left\_motor.Stop();

}

void check\_x\_plus(float x\_coordinate) //using RPS while robot is in the +x direction

{

//check whether the robot is within an acceptable range

while(RPS.X() < x\_coordinate - 1 || RPS.X() > x\_coordinate + 1)

{

if(RPS.X() > x\_coordinate)

{

//pulse the motors for a short duration in the correct direction

LCD.Write(RPS.X());

move\_forward(-40, 3);

}

else if(RPS.X() < x\_coordinate)

{

//pulse the motors for a short duration in the correct direction

LCD.Write(RPS.X());

move\_forward(30, 3);

}

}

while(RPS.X() < x\_coordinate - .2 || RPS.X() > x\_coordinate + .2)

{

if(RPS.X() > x\_coordinate)

{

//pulse the motors for a short duration in the correct direction

LCD.Write(RPS.X());

move\_forward(-40, 1);

}

else if(RPS.X() < x\_coordinate)

{

//pulse the motors for a short duration in the correct direction

LCD.Write(RPS.X());

move\_forward(30, 1);

}

}

}

void check\_x\_minus(float x\_coordinate) //using RPS while robot is in the +x direction

{

//check whether the robot is within an acceptable range

while(RPS.X() < x\_coordinate - 1 || RPS.X() > x\_coordinate + 1)

{

if(RPS.X() > x\_coordinate)

{

//pulse the motors for a short duration in the correct direction

LCD.Write(RPS.X());

move\_forward(30, 3);

}

else if(RPS.X() < x\_coordinate)

{

//pulse the motors for a short duration in the correct direction

LCD.Write(RPS.X());

move\_forward(-40, 3);

}

}

while(RPS.X() < x\_coordinate - .2 || RPS.X() > x\_coordinate + .2)

{

if(RPS.X() > x\_coordinate)

{

//pulse the motors for a short duration in the correct direction

LCD.Write(RPS.X());

move\_forward(30, 1);

}

else if(RPS.X() < x\_coordinate)

{

//pulse the motors for a short duration in the correct direction

LCD.Write(RPS.X());

move\_forward(-40, 1);

}

}

}

void check\_y\_plus(float y\_coordinate) //using RPS while robot is in the +x direction

{

//check whether the robot is within an acceptable range

while(RPS.Y() < y\_coordinate - 1 || RPS.Y() > y\_coordinate + 1)

{

if(RPS.Y() > y\_coordinate)

{

//pulse the motors for a short duration in the correct direction

LCD.Write(RPS.Y());

move\_forward(-40, 3);

}

else if(RPS.Y() < y\_coordinate)

{

//pulse the motors for a short duration in the correct direction

LCD.Write(RPS.Y());

move\_forward(30, 3);

}

}

while(RPS.Y() < y\_coordinate - .2 || RPS.Y() > y\_coordinate + .2)

{

if(RPS.Y() > y\_coordinate)

{

//pulse the motors for a short duration in the correct direction

LCD.Write(RPS.Y());

move\_forward(-40, 1);

}

else if(RPS.Y() < y\_coordinate)

{

//pulse the motors for a short duration in the correct direction

LCD.Write(RPS.Y());

move\_forward(30, 1);

}

}

}

void check\_y\_minus(float y\_coordinate) //using RPS while robot is in the +x direction

{

//check whether the robot is within an acceptable range

while(RPS.Y() < y\_coordinate - 1 || RPS.Y() > y\_coordinate + 1)

{

if(RPS.Y() < y\_coordinate)

{

//pulse the motors for a short duration in the correct direction

LCD.Write(RPS.Y());

move\_forward(-40, 3);

Sleep(50);

}

else if(RPS.Y() > y\_coordinate)

{

//pulse the motors for a short duration in the correct direction

LCD.Write(RPS.Y());

move\_forward(30, 3);

Sleep(50);

}

}

while(RPS.Y() < y\_coordinate - .2 || RPS.Y() > y\_coordinate + .2)

{

if(RPS.Y() < y\_coordinate)

{

//pulse the motors for a short duration in the correct direction

LCD.Write(RPS.Y());

move\_forward(-30, 1);

Sleep(50);

}

else if(RPS.Y() > y\_coordinate)

{

//pulse the motors for a short duration in the correct direction

LCD.Write(RPS.Y());

move\_forward(25, 1);

Sleep(50);

}

}

}

void check\_heading(float heading) //using RPS

{

while(RPS.Heading() < heading - 1 || RPS.Heading() > heading + 1)

{

if(RPS.Heading() > heading)

{

//pulse the motors for a short duration in the correct direction

LCD.Write(RPS.Heading());

turn\_right(60, 1);

Sleep(0.1);

}

else if(RPS.Heading() < heading)

{

//pulse the motors for a short duration in the correct direction

LCD.Write(RPS.Heading());

turn\_left(60, 1);

Sleep(0.1);

}

}

}

void check\_ramp\_plus()

{

while (RPS.Y() < 41)

{

move\_forward(70,3);

}

}

void check\_ramp\_minus()

{

while (RPS.Y() < 41)

{

move\_backward(70,3);

}

}

void red\_button\_press()

{

int n = 0, red;

LCD.WriteLine("Trying to press red button");

servo.SetDegree(180);

move\_forward(50,14);

Sleep(500);

move\_backward(50,14);

//Check to see if the button is pressed

red = RPS.RedButtonPressed();

//If the button was not pressed, then try 3 more times

while (red = 0)

{

//Check coordinates and adjust if needed

check\_heading(315.0);

move\_forward(50,14);

Sleep(500);

move\_backward(50,14);

//Check again to see if the button has been pressed

red = RPS.RedButtonPressed();

n = n + 1;

if (n = 3)

{

red = 1;

} //End if

} //End while

} //End red button press

void blue\_button\_press()

{

int n = 0, blue;

LCD.WriteLine("Trying to press blue button");

servo.SetDegree(45);

move\_forward(50,14);

Sleep(500);

move\_backward(50,14);

blue = RPS.BlueButtonPressed();

while (blue = 0)

{

check\_heading(315.0);

move\_forward(50,14);

Sleep(500);

move\_backward(50,14);

blue = RPS.BlueButtonPressed();

n = n + 1;

if (n = 3)

{

blue = 1;

} //End if

} //End while

} //End blue button press

void white\_button\_press()

{

int n = 0, white;

LCD.WriteLine("Trying to press white button");

servo.SetDegree(135);

move\_forward(50,14);

Sleep(500);

move\_backward(50,14);

white = RPS.WhiteButtonPressed();

while (white = 0)

{

check\_heading(315.0);

move\_forward(50,14);

Sleep(500);

move\_backward(50,14);

white = RPS.WhiteButtonPressed();

n = n + 1;

if (n = 3)

{

white = 1;

} //End if

} //End while

} //End white button press

void button\_task()

{

servo.SetMin(500);

servo.SetMax(2471);

LCD.WriteLine("In Button Task function");

if (RPS.RedButtonOrder() == 1)

{

LCD.WriteLine("Red 1");

red\_button\_press();

} //End if

else if (RPS.BlueButtonOrder() == 1)

{

LCD.WriteLine("Blue 1");

blue\_button\_press();

} //End else if

else if (RPS.WhiteButtonOrder() == 1)

{

LCD.WriteLine("White 1");

white\_button\_press();

} //End else if

if (RPS.RedButtonOrder() == 2)

{

LCD.WriteLine("Red 2");

red\_button\_press();

} //End else if

else if (RPS.BlueButtonOrder() == 2)

{

LCD.WriteLine("Blue 2");

blue\_button\_press();

} //End else if

else if (RPS.WhiteButtonOrder() == 2)

{

LCD.WriteLine("White 2");

white\_button\_press();

} //End else if

if (RPS.RedButtonOrder() == 3)

{

LCD.WriteLine("Red 3");

red\_button\_press();

} //End else if

else if (RPS.BlueButtonOrder() == 3)

{

LCD.WriteLine("Blue 3");

blue\_button\_press();

} //End else if

else if (RPS.WhiteButtonOrder() == 3)

{

LCD.WriteLine("White 3");

white\_button\_press();

} //End else if

} //End function button task

void salt\_up(float start\_heading)

{

scoop.SetPercent(50);

Sleep(0.6);

scoop.SetPercent(0);

move\_forward(35,55);

scoop.SetPercent(-70);

Sleep(1.0);

scoop.SetPercent(0);

Sleep(0.2);

move\_backward(35,30);

}

void crank\_turn()

{

int x = 0;

while(x==0)

{

LCD.Write(cds.Value());

if (cds.Value()<0.1)

{

LCD.WriteLine(cds.Value());

crank.SetPercent(-75);

Sleep(1.3);

crank.SetPercent(0);

x = 1;

}

else if (cds.Value()>0.1)

{

LCD.WriteLine(cds.Value());

crank.SetPercent(75);

Sleep(1.3);

crank.SetPercent(0);

x = 1;

}

}

}

int main(void)

{

servo.SetDegree(45);

ButtonBoard buttons( FEHIO::Bank3 );

//Set the LCD screen color

LCD.Clear( FEHLCD::Scarlet );

LCD.SetBackgroundColor(FEHLCD::Scarlet);

LCD.SetFontColor( FEHLCD::Gray );

//Call this function to initialize the RPS to a course

RPS.InitializeMenu();

//Press middle button to begin

LCD.WriteLine("Press the middle button to begin");

//Wait for middle button to be pressed

while(cds.Value() > 0.5);

//Wait for middle button to be unpressed

float x\_real=0,xadjust=0;//x\_real will be established once a qr code is attached to the robot

float t\_real=0,yadjust=0;//y\_real will be established once a qr code is attached to the robot

float motor\_percent = 50; //Input power level here

float counts\_per\_inch = 14.14; //Input theoretical counts here

float start\_heading=RPS.Heading();

float start\_x = RPS.X();

float start\_y = RPS.Y();

check\_heading(start\_heading);

//alignmentcode would go here

//saltbag ssuff would go here too

move\_forward(50,90);

Sleep(500);

LCD.WriteLine("Im broke fix me");

check\_y\_minus(start\_y - 10);

turn\_left(70,40);

Sleep(500);

check\_heading(start\_heading + 90);

move\_forward(50,90);

Sleep(500);

check\_x\_plus(start\_x + 10);

turn\_right(70,40);

check\_heading(start\_heading - 12);

move\_backward(60,200);

check\_ramp\_minus();

check\_y\_minus(start\_y + 21);

turn\_left(70,20);

check\_heading(start\_heading + 10);

move\_backward(50,50);

check\_y\_minus(start\_y + 24);

check\_heading(start\_heading);

move\_backward(50,10);

check\_y\_minus(start\_y + 27);

crank\_turn();

// turn\_left(70,80);

// Sleep(500);

// check\_heading(start\_heading + 180);

// move\_forward(50,212.1);

// Sleep(500);

// check\_ramp();

// check\_y\_plus(49.0);

// turn\_left(70,80);

// Sleep(500);

// check\_heading(start\_heading + 270);

// move\_forward(50,180);

// Sleep(500);

// move\_backward(50,150);

// check\_x\_minus(29.3);

// turn\_right(70,40);

// Sleep(500);

// check\_heading(start\_heading + 250);

// move\_forward(50,142);

// Sleep(500);

// check\_x\_minus(22.5);

// turn\_right(70,30);

// Sleep(500);

// check\_heading(start\_heading + 225);

// move\_forward(50,84);

// Sleep(500);

// check\_x\_minus(14.0);

// button\_task();

servo.SetDegree(45);

}