//\*\*\*\*\*Xboat\_3\*\*\*\*\*\*\*\*\*\*\*\*

/\*This is the operating program for the model of the RobotX boat.

It is based upon the Arbotix-M controller PCB and Arduino Software.

It assumes that there is a program running on a PC, likely written in LabVIEW.

The communication between the controller and the PC is done via XBEE devices.

They are set to 38400 baud, 8, N, 1, N and have a preset protocol.

The PC initiates all communication and expects a return confirming operation.

\*/

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <Servo.h> //for RC servos

#include <ax12.h> //for Dynamixel servos

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*GLOBALS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

boolean ledON; //used to toggle the led

unsigned long newtime; //the time at the start of loop()

unsigned long timeperiod; //time for loop() in milliseconds.

int estop\_val; //etop\_val == LOW is stopped

//create RC servo objects

Servo starboard\_motor;

Servo port\_motor;

Servo right\_rudder;

Servo left\_rudder;

int sm\_pos = 1500; //starboard motor position 0 to 2500

int pm\_pos = 1500; //port motor position

int rr\_pos = 1500; //right rudder position

int lr\_pos = 1500; //left rudder position

int cp\_pos = 1500; //camera pan position

int h\_code = 1; //Status code from PC 1 == OK, 100 == ESTOP

int ESTOP = 100;

float bat\_volt = 8.00; //Lipo battery voltage

float ir\_volt = 1.00; //IR distance voltage

const int buffer\_size = 40;

char readbuffer[buffer\_size];

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*MAPPING OF CONTROLLER BOARD PINS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*MAPPING OF DIO PINS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int led\_pin = 0; //USER LED on the ArbotiX Robocontroller, the blinking light

int ESTOP\_pin = 23; //manual switch on the model...stops all servos

int blink\_pin = 1; //running LED

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*MAPPING OF ANALOG INPUT PINS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int bat\_pin = 1; //Lipo battery voltage

int ir\_pin = 2; //Sharp IR distance device

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*MAPPING OF RC SERVO DEVICES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int sm\_pin = 12; //starboard motor speed

int pm\_pin = 13; //port motor speed

int rr\_pin = 14; //right rudder position

int lr\_pin = 15; //left rudder position

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*MAPPING OF DYNAMIXEL SERVO DEVICES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int cp\_num = 1; //the camera panning servo number

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*THE SETUP LOOP\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void setup()

{

// start serial port at 38400 bps:

Serial.begin(38400);

// wait to connect

delay(1000);

//setup for the Dynamixel serovs

SetPosition(1,0); //set the position of servo # 1 to '0'

//setup up the RC servos

starboard\_motor.attach(sm\_pin);

starboard\_motor.writeMicroseconds(sm\_pos);

port\_motor.attach(pm\_pin);

port\_motor.writeMicroseconds(pm\_pos);

right\_rudder.attach(rr\_pin);

right\_rudder.writeMicroseconds(rr\_pos);

left\_rudder.attach(lr\_pin);

left\_rudder.writeMicroseconds(lr\_pos);

//setup input pins

pinMode(ESTOP\_pin, INPUT); // set pin to input

digitalWrite(ESTOP\_pin, HIGH); // turn on pullup resistors, 20k resitor to 5VDC on input pin

//setup output pins

pinMode(led\_pin, OUTPUT); // initialize the digital pin as an output for onboard LED

pinMode(blink\_pin, OUTPUT); //output for remote blinking LED

//setup the timing and blinky led

ledON = false; //used to toggle the board led with loop timer

timeperiod = 100; //loop time in milliseconds

}// end setup

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*THE MAIN PROGRAM LOOP\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void loop()

{

//use the computer clock to create a timed loop, read the present time to start the timer

newtime = millis();

//read the value of the ESTOP switch, value is LOW when button is pressed, otherwise HIGH

estop\_val = digitalRead(ESTOP\_pin);

//Read the sensor values

Read\_sensors();

//Read the data bytes from the PC

int numread = 0;

numread = Read\_Command\_String();

//send a response in bytes to the PC if one has been read

if (numread > 0)

{

Send\_Return\_String(numread);

}

//Parse the Command\_String

Parse\_command\_string();

//Run the arbiter

//Move the motors

Move\_the\_motors();

//Move the rudders

Move\_the\_rudders();

//Move the camera

Move\_the\_camera();

//use computer clock to time the loop

while(millis() < (newtime + timeperiod) )

{

//wait until time elaspes

}

//change the state of the on board LED

Toggle\_led();

}// end void

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int Read\_Command\_String(void) //returns number of bytes found

{

//read the string as an array of type char

int num = 0; //number of bytes read

if ( Serial.available() > 0)

{

//read until the '\n' is found

num = Serial.readBytesUntil('\n', readbuffer, buffer\_size);

}

return num;

}//end Read\_Command\_String

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void Send\_Return\_String(int numread)

{

//send the response as a byte array

byte returnbuffer[3];

//if something has been sent, send a return

if (numread > 0)

{

returnbuffer[0] = 0x24; //$

returnbuffer[1] = 0x52; //R

returnbuffer[2] = 0x49; //I

Serial.write(returnbuffer, 3); //return the bytes read

Serial.print(ir\_volt,3); //sent the IR distance sensor

Serial.write(0x0A); //a newline, \n

}

return;

}// end Send\_Return\_String

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void Toggle\_led(void)

{

//toggle the led on the board

if (ledON == false)

{

ledON = true;

digitalWrite(led\_pin, HIGH); // turn the LED on (HIGH is the voltage level)

digitalWrite(blink\_pin, HIGH); //turn external LED on

}

else

{

ledON = false;

digitalWrite(led\_pin, LOW); // turn the LED off by making the voltage LOW

digitalWrite(blink\_pin, LOW); //turn external LED off

}

}//end Toggle\_led

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void Move\_the\_motors(void)

{

//move the motors which respond to RC servo microsecond values

//the motor controller is assumed to be a Marine 15

//check ESTOP, value is low when button is pressed

//and the passed stop value h\_code

if (estop\_val == LOW || h\_code == ESTOP)

{

//stop the motors

starboard\_motor.writeMicroseconds(1470); //tuned to motor??????????

port\_motor.writeMicroseconds(1500);

}

else

{

//move the motors to the passed positions

starboard\_motor.writeMicroseconds(sm\_pos);

port\_motor.writeMicroseconds(pm\_pos);

}

}//end Move\_the\_motors

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void Read\_sensors(void)

{

//This assumes the 5.0 VDC voltage reference is being used on the Arbotix-m board, the default

//read the Sharp IR distance sensor

int ir\_value;

ir\_value = analogRead(ir\_pin);

//convert to volts

ir\_volt = (float)ir\_value \* 5.0 / 1024.0;

}//end Read\_sensors

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void Parse\_command\_string(void)

{

//the following converts the 4 char number values into integers

//the readbuffer is assumed to be $CS----P----R----L----C----H----'\n' 34 bytes

char dummy[4];

//for starboard motor

dummy[0] = readbuffer[3];

dummy[1] = readbuffer[4];

dummy[2] = readbuffer[5];

dummy[3] = readbuffer[6];

sm\_pos = atoi(dummy); //convert char/byte array into interger

//for port motor

dummy[0] = readbuffer[8];

dummy[1] = readbuffer[9];

dummy[2] = readbuffer[10];

dummy[3] = readbuffer[11];

pm\_pos = atoi(dummy);

//for right rudder

dummy[0] = readbuffer[13];

dummy[1] = readbuffer[14];

dummy[2] = readbuffer[15];

dummy[3] = readbuffer[16];

rr\_pos = atoi(dummy);

//for left rudder

dummy[0] = readbuffer[18];

dummy[1] = readbuffer[19];

dummy[2] = readbuffer[20];

dummy[3] = readbuffer[21];

lr\_pos = atoi(dummy);

//for camera pan

dummy[0] = readbuffer[23];

dummy[1] = readbuffer[24];

dummy[2] = readbuffer[25];

dummy[3] = readbuffer[26];

cp\_pos = atoi(dummy);

//for status

dummy[0] = readbuffer[28];

dummy[1] = readbuffer[29];

dummy[2] = readbuffer[30];

dummy[3] = readbuffer[31];

h\_code = atoi(dummy);

}//end Parse\_commmand\_string

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void Move\_the\_camera(void)

{

//the camera is panned using a Dynamixel servo motor AX-12

//below are the values of m and b in y = m\*x + b

//this moves the range 500 to 2500 into the range 0 to 1023 required by the Dynamixel servo

float m = 0.51;

float b = -255.0;

float cp\_value;

//calculate the camera position of 0 to 1023;

cp\_value = m \* (float)cp\_pos + b;

//check ESTOP, value is low when button is pressed

//and the passed stop value h\_code

if (estop\_val == LOW || h\_code == ESTOP)

{

//leave the camera alone

}

else

{

//move the camera to the passed positions

SetPosition(1, (int)cp\_value);

}

}//end Move\_the\_camera

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void Move\_the\_rudders(void)

{

//the rudders are assumed to be operated by RC servos

//check ESTOP, value is low when button is pressed

//and the passed stop value h\_code

if (estop\_val == LOW || h\_code == ESTOP)

{

//leave the camera alone

}

else

{

//move the rudders

right\_rudder.writeMicroseconds(rr\_pos);

left\_rudder.writeMicroseconds(lr\_pos);

}

}//end Move\_the\_rudders

//\*\*\*\*\*\*\*\*\*\*\*\*\* end of program\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*