/\*

 \* Final

 \*/

//Fer,AJ,Daniel

#include <Arduino.h>

#include <wpi-32u4-lib.h>

#include <IRdecoder.h>

#include <ir\_codes.h>

#include <Chassis.h>// TODO, Section, 4.2: Add line to include Chassis.h

#include <Rangefinder.h>

#define LED\_YELLOW 13

#define LED\_RED 17

#define LED\_GREEN 30

int baseSpeed = 15; //cm/second

float turneffort, error, K\_p = .1;

float n\_error;

bool construction = 0;

int globalstep = 0;

int ubersteps = 0;

// Sets up the IR receiver/decoder object

const uint8\_t IR\_DETECTOR\_PIN = 1;

IRDecoder decoder(IR\_DETECTOR\_PIN);

//distance between tires first one, angle change third

Chassis chassis(7,1440,13.5);// TODO, Section 6.2: Adjust parameters to better match actual motion

Rangefinder rangefinder(11,4);

enum UBER\_ROUTE {ONE\_LONG\_DEL, TWO\_SHORT\_DEL, TWO\_LONG\_DEL, THREE\_SHORT\_DEL, THREE\_LONG\_DEL,

                ONE\_LONG\_START, TWO\_SHORT\_START, TWO\_THREE\_LONG\_START, THREE\_SHORT\_START,

                FREE\_PICKUP, IGNORE};

//one is 8cm

//two is 4cm

//three is 0cm

UBER\_ROUTE ROUTE;

void setLED(bool value) // A helper function for debugging

{

  Serial.println("setLED()");

  digitalWrite(LED\_YELLOW, value);

}

// Defines the robot states

enum ROBOT\_STATE {ROBOT\_IDLE, ROBOT\_DRIVE\_FOR, ROBOT\_LINING, ROBOT\_UBER, ROBOT\_WAIT\_UNTIL};

ROBOT\_STATE robotState = ROBOT\_IDLE;

void idle(void) // idle() stops the motors

{

  Serial.println("idle()");

  setLED(LOW);

  chassis.idle();  // TODO, Section 4.2: Uncomment call to chassis.idle() to stop the motors

  if(robotState == ROBOT\_WAIT\_UNTIL)

  {

    if(construction == 1)

    {

        //route(construction);

        ROUTE = TWO\_THREE\_LONG\_START;

        construction = false;

    }

    else

    {

      globalstep++;

      robotState = ROBOT\_UBER;

    }

  }

  else

  {

    robotState = ROBOT\_IDLE;   //set state to idle

  }

}

void beginLineFollowing(void)

{

  Serial.println("beginLineFollowing()");

  setLED(HIGH);

  //Create a search line state

  robotState = ROBOT\_LINING;

}

void handleLineFollow()

{

  Serial.println("handleLineFollow()");

  int leftLineSensorReading = analogRead(LEFT\_LINE\_SENSE);

  int middleLineSensorReading = analogRead(MIDDLE\_LINE\_SENSE);

  int rightLineSensorReading = analogRead(RIGHT\_LINE\_SENSE);

  // Serial.print('\t');

  // Serial.print(leftLineSensorReading);

  // Serial.print('\t'); //print a TAB character to make the output prettier

  // Serial.print(middleLineSensorReading);

  // Serial.print('\t');

  // Serial.print(rightLineSensorReading);

  // delay(100);

  K\_p = .1;

  error =  rightLineSensorReading - leftLineSensorReading;

  //n\_error = error \* K\_p;

  turneffort = error \* K\_p ;

  if (middleLineSensorReading > 460) {

    if(rightLineSensorReading < 460 && leftLineSensorReading < 460) {

        //code for drive straight over line

          turneffort = error \* K\_p;

          chassis.setTwist(baseSpeed, turneffort);

    } else if (rightLineSensorReading > 460 && leftLineSensorReading < 460){

        //code for right turn

        Serial.println("Turning Right");

          turneffort = error \* K\_p;

          chassis.setTwist(baseSpeed, turneffort);

    } else if (rightLineSensorReading < 460 && leftLineSensorReading > 460){

        //code for left turn

        Serial.println("Turning Left");

        turneffort = error \* K\_p;

        chassis.setTwist(baseSpeed, turneffort);

    }

    else if (rightLineSensorReading > 460 && leftLineSensorReading > 460){

        //intersection hit; robot stops

        idle();

    } //Now, to handle if the middle sensor isn't over the line but one of the side sensors is:

  }

  else if(middleLineSensorReading < 460 && rightLineSensorReading > 460){

      //code for hard right turn

        K\_p = .3;

        turneffort = (error) \* K\_p;

        chassis.setTwist(baseSpeed, turneffort);

  } else if(middleLineSensorReading < 460 && leftLineSensorReading > 460){

      //code for hard left turn

        K\_p = .3;

        turneffort = (error) \* K\_p;

        chassis.setTwist(baseSpeed, turneffort);

  } else if(middleLineSensorReading < 460 && rightLineSensorReading < 460 && leftLineSensorReading < 460){

      //in the scenario that the robot completely leaves the line...

      //insert Daniel's code for line finding here

      //i assume the robot's either going to rotate until it finds the line, or

      //it'll back up to retrace its steps and find it.

  }

}

void travelOnLine()

{

  Serial.println("travelOnLine()");

  int leftLineSensorReading = analogRead(LEFT\_LINE\_SENSE);

  int middleLineSensorReading = analogRead(MIDDLE\_LINE\_SENSE);

  int rightLineSensorReading = analogRead(RIGHT\_LINE\_SENSE);

  // Serial.print('\t');

  // Serial.print(leftLineSensorReading);

  // Serial.print('\t'); //print a TAB character to make the output prettier

  // Serial.print(middleLineSensorReading);

  // Serial.print('\t');

  // Serial.print(rightLineSensorReading);

  // Serial.print('\n');

  // delay(100);

  K\_p = .1;

  error =  rightLineSensorReading - leftLineSensorReading;

  turneffort = error \* K\_p ;

  if (rightLineSensorReading > 460 && leftLineSensorReading > 460)

  {

    //intersection hit; robot stops

    chassis.idle();

    Serial.println("chassis intersection");

    globalstep++;

  }

  else

  {

    chassis.setTwist(baseSpeed, turneffort);

  }

}

// bool intersection()

// {

//   int leftLineSensorReading = analogRead(LEFT\_LINE\_SENSE);

//   int rightLineSensorReading = analogRead(RIGHT\_LINE\_SENSE);

//   bool retVal = true;

//   bool rightsensor = rightLineSensorReading > 400 ? true : false;

//   bool leftsensor = leftLineSensorReading > 400 ? true : false;

//   if(rightsensor && leftsensor)

//   {

//     retVal = false;

//     //chassis.idle();

//   }

//   return retVal;

// }

/\*

 \* This is the standard setup function that is called when the board is rebooted

 \* It is used to initialize anything that needs to be done once.

 \*/

void setup()

{

  // This will initialize the Serial at a baud rate of 115200 for prints

  // Be sure to set your Serial Monitor appropriately

  Serial.begin(115200);

  chassis.init(); // TODO, Section 4.2: Initialize the chassis (which also initializes the motors)

  chassis.setMotorPIDcoeffs(3, 0.3);// TODO, Section 5.1: Adjust the PID coefficients

  idle();

  // Initializes the IR decoder

  decoder.init(); //Question 2

  Serial.println("/setup()");

  //Sensor

  pinMode(LEFT\_LINE\_SENSE, INPUT);

  pinMode(MIDDLE\_LINE\_SENSE, INPUT);

  pinMode(RIGHT\_LINE\_SENSE, INPUT);

  // Call init() to set up the rangefinder

  rangefinder.init();

}

// A helper command to drive a set distance

// At the start, it will take no arguments and we'll hardcode a motion

// TODO, Section 6.1 (but not before!): Edit the function definition to accept a distance and speed

void drive(float distance, float speed)

{

  setLED(HIGH);

  robotState = ROBOT\_DRIVE\_FOR;

  //chassis.setWheelSpeeds(30, 30);// TODO: In Section 4.2 and 5.1, add a call to chassis.setWheelSpeeds() to set the wheel speeds 2

  chassis.driveFor(distance,speed); // TODO: In Section 6.1, remove the call to setWheelSpeeds() and add a call to chassis.driveFor()

}

// A helper function to turn a set angle

void turn(float ang, float speed)

{

  setLED(HIGH);

  robotState = ROBOT\_DRIVE\_FOR;

  chassis.turnFor(ang,speed);// TODO, Section 6.1: Make a call to chassis.turnFor()

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* UBER \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void turnLeft()

{

  robotState = ROBOT\_WAIT\_UNTIL;

  Serial.println("turning left");

  chassis.turnFor(90.0,100.0);

}

void turnRight()

{

  robotState = ROBOT\_WAIT\_UNTIL;

  Serial.println("turning right");

  chassis.turnFor(-90.0,100.0);

}

void forward()

{

  robotState = ROBOT\_WAIT\_UNTIL;

  Serial.println("forward");

  chassis.driveFor(6.0,12.0);

}

void turn180()

{

  robotState = ROBOT\_WAIT\_UNTIL;

  Serial.println("turning 180");

  chassis.turnFor(180.0,100.0);

}

void beginUber(void)

{

  robotState = ROBOT\_UBER;

  ROUTE = ONE\_LONG\_DEL;

}

/\* This function is to take the robot from the drop-off, back to the intersection before the pickup \*/

// void backtostart()

// {

// switch(START)

//     {

//       case ONE\_LONG\_START:

//         switch(globalstep)

//         {

//           case 0: travelOnLine(); break;

//           case 1: forward(); break;

//           case 2: turnLeft(); break;

//           case 3: travelOnLine(); break;

//           case 4: forward(); break;

//           case 5: turnRight(); break;

//           case 6: travelOnLine(); break;

//           case 7: forward(); break;

//           case 8: turnLeft(); break;

//           case 9: travelOnLine(); break;

//           default: chassis.idle(); break;

//         }

//         break;

//       case TWO\_THREE\_LONG\_START:

//       switch(globalstep)

//         {

//           case 0: travelOnLine(); break;

//           case 1: forward(); break;

//           case 2: turnRight(); break;

//           case 3: travelOnLine(); break;

//           case 4: forward(); break;

//           case 5: travelOnLine(); break;

//           case 6: turnRight(); break;

//           case 7: travelOnLine(); break;

//           case 8: forward(); break;

//           case 9: turnRight(); break;

//           case 10: travelOnLine(); break;

//           case 11: forward(); break;

//           case 12: turnLeft(); break;

//           case 13: travelOnLine(); break;

//           default: chassis.idle(); break;

//         }

//       break;

//     case TWO\_SHORT\_START:

//       switch(globalstep)

//       {

//         case 0: travelOnLine(); break;

//         case 1: forward(); break;

//         case 2: travelOnLine(); break;

//         //sense box/construction

//         case 3: forward(); break;

//         case 4: travelOnLine(); break;

//         default: chassis.idle(); break;

//       }//switch

//       break;

//     case THREE\_SHORT\_START:

//       switch(globalstep)

//       {

//         case 0: travelOnLine(); break;

//         case 1: forward(); break;

//         case 2: turnRight(); break;

//         case 3: travelOnLine(); break;

//         //sense box/construction

//         case 4: forward(); break;

//         case 5: travelOnLine(); break;

//         default: chassis.idle(); break;

//       }//switch

//       break;

//     default:break;

//     }//switch

// }

// /\* all the decisions \*/

void route()

{

  switch(ROUTE)

    {

      case ONE\_LONG\_DEL:

        switch(globalstep)

        {

          //pickup

          case 0: travelOnLine(); break;

          case 1: forward(); break;

          case 2: turnRight(); break;

          case 3: travelOnLine(); break;

          case 4: forward(); break;

          case 5: turnLeft(); break;

          case 6: travelOnLine(); break;

          case 7: forward(); break;

          case 8: turnRight(); break;

          case 9: travelOnLine(); break;

          //dropoff

          case 10: turn180(); break;

          default: chassis.idle(); break;

        }

        break;

      case TWO\_LONG\_DEL:

      switch(globalstep)

        {

          case 0: travelOnLine(); break;

          case 1: forward(); break;

          case 2: turnRight(); break;

          case 3: travelOnLine(); break;

          case 4: forward(); break;

          case 5: turnLeft(); break;

          case 6: travelOnLine(); break;

          case 7: forward(); break;

          case 8: turnLeft(); break;

          case 9: travelOnLine(); break;

          case 10: forward(); break;

          case 11: travelOnLine(); break;

          case 12: forward(); break;

          case 13: turnRight(); break;

          case 14: travelOnLine(); break;

          case 15: turn180(); break; //dropoff

          case 16: travelOnLine(); break;

          //backtostart check intersection

          default: chassis.idle(); break;

        }

      break;

      case THREE\_LONG\_DEL:

      switch(globalstep)

        {

          case 0: travelOnLine(); break;

          case 1: forward(); break;

          case 2: turnRight(); break;

          case 3: travelOnLine(); break;

          case 4: forward(); break;

          case 5: turnLeft(); break;

          case 6: travelOnLine(); break;

          case 7: forward(); break;

          case 8: turnLeft(); break;

          case 9: travelOnLine(); break;

          case 10: forward(); break;

          case 11: travelOnLine(); break;

          case 12: forward(); break;

          case 13: travelOnLine(); break;

          //adjustments & pickup

          default: chassis.idle(); break;

        }

      break;

    case TWO\_SHORT\_DEL:

      switch(globalstep)

      {

        case 0: travelOnLine(); break;

        case 1: forward(); break;

        case 2: travelOnLine(); break;

        case 3: forward(); break;

        case 4: travelOnLine(); break;

        default: chassis.idle(); break;

      }//switch

      break;

    case THREE\_SHORT\_DEL:

      switch(globalstep)

      {

        case 0: travelOnLine(); break;

        case 1: forward(); break;

        case 2: travelOnLine(); break;

        case 3: forward(); break;

        case 4: turnLeft(); break;

        case 5: travelOnLine(); break;

        default: chassis.idle(); break;

      }//switch

      break;

    case ONE\_LONG\_START:

        switch(globalstep)

        {

          case 0: travelOnLine(); break;

          case 1: forward(); break;

          case 2: turnLeft(); break;

          case 3: travelOnLine(); break;

          case 4: forward(); break;

          case 5: turnRight(); break;

          case 6: travelOnLine(); break;

          case 7: forward(); break;

          case 8: turnLeft(); break;

          case 9: travelOnLine(); break;

          default: chassis.idle(); break;

        }

        break;

case TWO\_THREE\_LONG\_START:

      switch(globalstep)

        {

          case 0: travelOnLine(); break;

          case 1: forward(); break;

          case 2: turnRight(); break;

          case 3: travelOnLine(); break;

          case 4: forward(); break;

          case 5: travelOnLine(); break;

          case 6: turnRight(); break;

          case 7: travelOnLine(); break;

          case 8: forward(); break;

          case 9: turnRight(); break;

          case 10: travelOnLine(); break;

          case 11: forward(); break;

          case 12: turnLeft(); break;

          case 13: travelOnLine(); break;

          default: chassis.idle(); break;

        }

      break;

    case TWO\_SHORT\_START:

      switch(globalstep)

      {

        case 0: travelOnLine(); break;

        case 1: forward(); break;

        case 2: travelOnLine(); break;

        //sense box/construction

        case 3: forward(); break;

        case 4: travelOnLine(); break;

        default: chassis.idle(); break;

      }//switch

      break;

    case THREE\_SHORT\_START:

      switch(globalstep)

      {

        case 0: travelOnLine(); break;

        case 1: forward(); break;

        case 2: turnRight(); break;

        case 3: travelOnLine(); break;

        //sense box/construction

        case 4: forward(); break;

        case 5: travelOnLine(); break;

        default: chassis.idle(); break;

      }//switch

      break;

    default:break;

    }//routeswitch

}

// void deliver()

// {

//   Serial.println("deliver() globalstep = " + String(globalstep));

//   switch(ROUTE)

//     {

//       case ONE\_LONG\_DEL:

//         switch(globalstep)

//         {

//           //pickup

//           case 0: travelOnLine(); break;

//           case 1: forward(); break;

//           case 2: turnRight(); break;

//           case 3: travelOnLine(); break;

//           case 4: forward(); break;

//           case 5: turnLeft(); break;

//           case 6: travelOnLine(); break;

//           case 7: forward(); break;

//           case 8: turnRight(); break;

//           case 9: travelOnLine(); break;

//           //dropoff

//           case 10: turn180(); break; //dropoff/servo();

//           default: chassis.idle(); break;

//         }

//         break;

//       case TWO\_LONG\_DEL:

//       switch(globalstep)

//         {

//           case 0: travelOnLine(); break;

//           case 1: forward(); break;

//           case 2: turnRight(); break;

//           case 3: travelOnLine(); break;

//           case 4: forward(); break;

//           case 5: turnLeft(); break;

//           case 6: travelOnLine(); break;

//           case 7: forward(); break;

//           case 8: turnLeft(); break;

//           case 9: travelOnLine(); break;

//           case 10: forward(); break;

//           case 11: travelOnLine(); break;

//           case 12: forward(); break;

//           case 13: turnRight(); break;

//           case 14: travelOnLine(); break;

//           case 15: turn180(); break; //dropoff

//           case 16: travelOnLine(); break;

//           //backtostart check intersection

//           default: chassis.idle(); break;

//         }

//       break;

//       case THREE\_LONG\_DEL:

//       switch(globalstep)

//         {

//           case 0: travelOnLine(); break;

//           case 1: forward(); break;

//           case 2: turnRight(); break;

//           case 3: travelOnLine(); break;

//           case 4: forward(); break;

//           case 5: turnLeft(); break;

//           case 6: travelOnLine(); break;

//           case 7: forward(); break;

//           case 8: turnLeft(); break;

//           case 9: travelOnLine(); break;

//           case 10: forward(); break;

//           case 11: travelOnLine(); break;

//           case 12: forward(); break;

//           case 13: travelOnLine(); break;

//           //adjustments & pickup

//           default: chassis.idle(); break;

//         }

//       break;

//     case TWO\_SHORT\_DEL:

//       switch(globalstep)

//       {

//         case 0: travelOnLine(); break;

//         case 1: forward(); break;

//         case 2: travelOnLine(); break;

//         case 3: forward(); break;

//         case 4: travelOnLine(); break;

//         default: chassis.idle(); break;

//       }//switch

//       break;

//     case THREE\_SHORT\_DEL:

//       switch(globalstep)

//       {

//         case 0: travelOnLine(); break;

//         case 1: forward(); break;

//         case 2: travelOnLine(); break;

//         case 3: forward(); break;

//         case 4: turnLeft(); break;

//         case 5: travelOnLine(); break;

//         default: chassis.idle(); break;

//       }//switch

//       break;

//     default:break;

//     }//switch

// }

void servo()

{

  Serial.println("Servo()...");

  //turn180();

  //pickup

  //dropoff

}

void ubermensch()

{

  Serial.println("ubermensch()");

  //start();

  // route();

  switch (ubersteps)

  {

  case 0: ROUTE = ONE\_LONG\_DEL; Serial.println("" + String(ROUTE)); route(); break;

  case 1: globalstep = 0; break;

  case 2: servo(); break;

  case 3: ROUTE = ONE\_LONG\_START; Serial.println("Switching route ->" + String(ROUTE));route(); break;

  case 4: globalstep = 0; break;

  case 5: servo();

  case 6: ROUTE = TWO\_LONG\_DEL; Serial.println("Switching route ->" + String(ROUTE)); route(); break;

  case 7: globalstep = 0; break;

  case 8: servo(); break;

  case 9: ROUTE = TWO\_SHORT\_START; route(); break;

  case 10: globalstep = 0; break;

  case 11: servo(); break;

  case 12: ROUTE = THREE\_LONG\_DEL; route(); break;

  case 13: globalstep = 0; break;

  case 14: ROUTE = THREE\_SHORT\_START; route(); break;

  case 15: globalstep = 0; break;

  default: break;

  }

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* UBER END \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// TODO, Section 6.1: Declare function handleMotionComplete(), which calls idle()

void handleMotionComplete()

{

    idle();

}

// Handles a key press on the IR remote

void handleKeyPress(int16\_t keyPress)

{

  Serial.println("Key: " + String(keyPress));

       if(keyPress == 9) // TODO, Section 3.2: add "emergency stop"

       {

          idle();

       }

  switch(robotState)

  {

    case ROBOT\_IDLE:

      if(keyPress == 5) // TODO, Section 3.2: Handle up arrow button

      {

        Serial.println("Driving Forward");

        drive(80,baseSpeed);

      }

      else if(keyPress == 13) //back arrow // TODO, Section 6.1: Handle remaining arrows

      {

        Serial.println("Driving Back");

        drive(-50,-(baseSpeed)); //distance, speed

      }

      else if(keyPress == 8) //left arrow

      {

        Serial.println("Turning Left");

        turn(90.0, 40.0);

      }

     else if(keyPress == 10) //right arrow

      {

        Serial.println("Turning Right");

        turn(-90.0, 40.0);

      }

      else if(keyPress == 4) //setup

      {

        Serial.println("Robot Lining");

        beginLineFollowing();

      }

      else if(keyPress == 14) //rewind/back

      {

        beginUber();

      }

      else if(keyPress == 12) //0 10+

      {

        globalstep = 0;

        ubersteps = 0;

        Serial.println("globalstep = " + String(globalstep));

        Serial.println("ubersteps = " + String(ubersteps));

      }

      break;

    case ROBOT\_LINING:

      if(keyPress == 0) // VOL -

      {

        baseSpeed -= 3;

        Serial.println("baseSpeed - 3 = " + String(baseSpeed));

      }

      else if(keyPress == 2) // VOL +

      {

         baseSpeed += 3;

         Serial.println("baseSpeed + 3 = " + String(baseSpeed));

      }

      break;

    case ROBOT\_UBER:

    //Pickup locations in order

      switch (keyPress)

      {

      case 12: globalstep = 0;ubersteps = 0; Serial.println("globalstep = " + String(globalstep)); break;

      //1 2 3

      case 16: ubersteps++; break;

      case 17:  break;

      case 18: Serial.println("saving..."); break;

      //4 5 6

      case 20: ROUTE = TWO\_LONG\_DEL;globalstep = 0; break;

      case 21: ROUTE = TWO\_SHORT\_START; globalstep = 0; break;

      case 22:           break;

      //7 8 9

      case 24: ROUTE = THREE\_LONG\_DEL;globalstep = 0; break;

      case 25: ROUTE = THREE\_SHORT\_START;globalstep = 0; break;

      case 26:           break;

      default: break;

      }

    break;

    default:

      break;

  }

}

/\*

 \* The main loop for the program. The loop function is repeatedly called

 \* after setup() is complete.

 \*/

void loop()

{

  // Checks for a key press on the remote

  int16\_t keyPress = decoder.getKeyCode(); // TODO, Section 3.1: Temporarily edit to pass true to getKeyCode()

  if(keyPress >= 0) handleKeyPress(keyPress);

  float rangefinderdist = rangefinder.getDistance();

  if(rangefinderdist < 10.00)

  {

    Serial.println("OBJECT INFRONT! " + String(rangefinderdist));

    construction = true;

    idle();

  }

  // A basic state machine

  switch(robotState)

  {

    case ROBOT\_DRIVE\_FOR:

      // TODO, Section 6.1: Uncomment to handle completed motion

      if(chassis.checkMotionComplete()) handleMotionComplete();

      break;

    case ROBOT\_LINING:

      handleLineFollow();

    break;

    case ROBOT\_UBER:

      ubermensch();

      //route();

      //deliver();

    break;

    case ROBOT\_WAIT\_UNTIL:

      if(chassis.checkMotionComplete()) handleMotionComplete();

    break;

    default:

      break;

  }

}