```
#include <Timers.h>

#include <Raptor.h>
#include <SPI.h>

/*************************************************************
  File:      RaptorBasics.ino
  Contents:  This program is a warmup for ME210 Lab 0, and
             serves as an introduction to event-driven programming
  Notes:     Target: Arduino Leonardo
             Arduino IDE version: 1.6.7

  History:
  when       who  what/why
  ----       ---  ------------------------------------------
  2016-01-09 MTP  program created
  2016-01-10 KN   Updated Raptorlib and RaptorProof to match
  2016-01-11 KLG  minor tweaks to make signed/unsigned consistent
 *************************************************************/

/*---------------Includes-----------------------------------*/

#include <Raptor.h>
#include <Timers.h>

/*---------------Module Defines-----------------------------*/

#define LIGHT_THRESHOLD    300 // *Choose your own thresholds* smaller at night
#define LINE_THRESHOLD    500 // *Choose your own thresholds*
#define EDGE_THRESHOLD      600

#define ONE_SEC            1000
#define TIME_INTERVAL     ONE_SEC

#define TIMER_0            0
#define TIMER_1            1

#define LEFT_TRIGGER      0x01
#define CENTER_TRIGGER      0x04
#define RIGHT_TRIGGER      0x10
/*---------------Module Function Prototypes-----------------*/
void checkGlobalEvents(void);
void handleMoveForward(void);
void handleMoveBackward(void);
unsigned char TestTimer0Expired(void);
void RespTimer0Expired(void);
unsigned char TestForKey(void);
void RespToKey(void);
unsigned char TestForLightOn(void);
void RespToLightOn(void);
unsigned char TestForLightOff(void);
void RespToLightOff(void);
unsigned char TestForFence(void);
void RespToFence(void);
unsigned char TestTimer0Expired(void);
void RespTimer0Expired(void);
void printLightLevel(void);
void printLineLevel(void);
unsigned char TestTimer1Expired(void);
unsigned char checkTimer1Active(void);
void checkState(void);
void handleLightOff(void);
void handleCenter(void);
void handleRight(void);
```

```
void handleLeft(void);
void handleForward(void);
void handleBack(void);
void handleTurn(void);
void moveForward(void);
void moveBackward(void);
void turnRight(void);
void turnLeft(void);
/*---------------State Definitions--------------------------*/
typedef enum {
   STATE_LIGHT_OFF, STATE_LEFT_DETECTED, STATE_RIGHT_DETECTED, STATE_CENTER_DETECTED,
   STATE_BACK, STATE_TURN, STATE_FORWARD
} States_t;

/*---------------Module Variables---------------------------*/
States_t state;
States_t prevState;
unsigned char isLEDOn;
unsigned char lineTouched;
unsigned char blackOut;
unsigned char nearLine;

/*---------------Raptor Main Functions----------------*/

void setup() {
  Serial.begin(9600);
  Serial.println("Hello, world!");
  state = STATE_FORWARD;
  prevState = state;
  isLEDOn = false;
  lineTouched = true;
  blackOut = false;
  nearLine = false;
  TMRArd_InitTimer(0, TIME_INTERVAL);
}

void loop() {
  checkGlobalEvents();
  checkState();
  printLineLevel();
  switch(state) {
    case STATE_LIGHT_OFF:
        handleLightOff();
        break;
    case STATE_CENTER_DETECTED:
        handleCenter();
        break;
    case STATE_RIGHT_DETECTED:
        handleRight();
        break;
    case STATE_LEFT_DETECTED:
        handleLeft();
        break;
    case STATE_BACK:
        handleBack();
        break;
    case STATE_TURN:
        handleTurn();
        break;
    case STATE_FORWARD:
        handleForward();
    default: //Should never get into an unhandled state
      Serial.println("What is this I do not even...");
  }
```

```
}

/*----------------Module Functions--------------------------*/
void checkGlobalEvents(void) {
  if (TestTimer0Expired()) RespTimer0Expired();
  if (TestForKey()) RespToKey();
}

void checkState(void){
    if(TestForLightOn()){
        if(state!=STATE_BACK){
            unsigned char triggerState=raptor.ReadTriggers(LINE_THRESHOLD);
            unsigned int leftIndicator = raptor.EdgeLeft();
            unsigned int rightIndicator = raptor.EdgeRight();
            if(triggerState&CENTER_TRIGGER){
                state=STATE_CENTER_DETECTED;
            }else if(leftIndicator<EDGE_THRESHOLD){
                state=STATE_LEFT_DETECTED;
            }else if(rightIndicator<EDGE_THRESHOLD){
                state=STATE_RIGHT_DETECTED;
            }else if(blackOut){
                state = prevState;
            }
        }else if(blackOut){
            state = prevState;
        }
    }else{
        state = STATE_LIGHT_OFF;
    }
}

void handleLightOff(void){
    if (blackOut==false){
        raptor.LeftMtrSpeed(0);
        raptor.RightMtrSpeed(0);
        blackOut = true;
        if (checkTimer1Active()) TMRArd_StopTimer(1);
    }
}

void handleLeft(void){
    if(lineTouched==false||blackOut) {
        turnLeft();
        lineTouched = true;
        blackOut = false;
    }
}
void handleRight(void){
        if(lineTouched==false||blackOut) {
        turnRight();
        lineTouched = true;
        blackOut = false;
    }
}

void handleCenter(void){
    TMRArd_InitTimer(TIMER_1, TIME_INTERVAL*5);
    moveBackward();
    lineTouched = false;
    state=STATE_BACK;
    prevState = STATE_BACK;
}

void handleBack(void){
```

```
    if(blackOut){
        blackOut = false;
        TMRArd_StartTimer(TIMER_1);
        moveBackward();
    }
    if (TestTimer1Expired()){
        TMRArd_InitTimer(TIMER_1, TIME_INTERVAL*4);
        turnRight();
        state=STATE_TURN;
        prevState = STATE_TURN;
    }
}

void handleTurn(void){
    if(blackOut){
        blackOut = false;
        TMRArd_StartTimer(TIMER_1);
        turnRight();
    }
    if(TestTimer1Expired()){
        moveForward();
        state=STATE_FORWARD;
        prevState = STATE_FORWARD;
    }
}
void handleForward(void){
    if(blackOut){
        blackOut = false;
        TMRArd_StartTimer(TIMER_1);
        moveForward();
    }
    if(lineTouched){
        moveForward();
        lineTouched = false;
    }
}

void moveBackward(void) {
  raptor.LeftMtrSpeed(-30);
  raptor.RightMtrSpeed(-30);
}

void moveForward(void) {
  raptor.LeftMtrSpeed(30);
  raptor.RightMtrSpeed(30);
}
void turnLeft(void){
    raptor.LeftMtrSpeed(0);
    raptor.RightMtrSpeed(25);
}
void turnRight(void){
    raptor.LeftMtrSpeed(25);
    raptor.RightMtrSpeed(0);
}

unsigned char TestTimer0Expired(void) {
  return (unsigned char)(TMRArd_IsTimerExpired(TIMER_0));
}

void RespTimer0Expired(void) {
  static int Time = 0;
  TMRArd_InitTimer(TIMER_0, TIME_INTERVAL);
  if(isLEDOn) {
    isLEDOn = false;
```

```
    raptor.RGB(RGB_OFF);
  } else {
    isLEDOn = true;
    raptor.RGB(RGB_WHITE);
  }
}


unsigned char TestTimer1Expired(void) {
  return (unsigned char)(TMRArd_IsTimerExpired(TIMER_1));
}

unsigned char checkTimer1Active(void) {
  return (unsigned char)(TMRArd_IsTimerActive(TIMER_1));
}

unsigned char TestForKey(void) {
  unsigned char KeyEventOccurred;
  KeyEventOccurred = Serial.available();
  return KeyEventOccurred;
}

void RespToKey(void) {
  unsigned char theKey;

  theKey = Serial.read();
  Serial.print(theKey);
  Serial.print(", ASCII=");
  Serial.println(theKey,HEX);
}

void printLightLevel(void){
  Serial.print("Light Level=");
  Serial.println(raptor.LightLevel());
}

void printLineLevel(void){
  Serial.print("\n Right_Line=");
  Serial.println(raptor.LineRight());
  Serial.print("Center_Line=");
  Serial.println(raptor.LineCenter());
  Serial.print("Left_Line=");
  Serial.println(raptor.LineLeft());
  Serial.print("Right_Edge=");
  Serial.println(raptor.EdgeRight());
  Serial.print("Left_Edge=");
  Serial.println(raptor.EdgeRight());
}

unsigned char TestForLightOn(void) {
  if((raptor.LightLevel() > LIGHT_THRESHOLD)){
    return (unsigned char)true;
  }else{
    return (unsigned char)false;
  }
}

void RespToLightOn(void) {
  raptor.RGB(RGB_WHITE);
}

unsigned char TestForLightOff(void) {
    if((raptor.LightLevel() < LIGHT_THRESHOLD)){
    return true;
```

```
  }else{
    return false;
  }
}
```