/\*

\* File: SmartSearch\_SubHSM.c

\* Author: Brian Naranjo & jquiamba

\* Date: 11/15/19

\*

\*/

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

\* MODULE #INCLUDE \*

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

#include "BOARD.h"

#include "AD.h"

#include "ES\_Configure.h"

#include "ES\_Framework.h"

#include "BotHSM.h"

#include "SmartSearch\_SubHSM.h"

#include "Motors.h"

#include "Tape.h"

#include <stdio.h>

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

\* MODULE #DEFINES \*

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

typedef enum {

INIT\_STATE,

WAITING,

AVOID\_CENTER\_TAPE,

AVOID\_LEFT\_TAPE,

AVOID\_RIGHT\_TAPE,

SWEEP\_LEFT,

SWEEP\_RIGHT,

RUN\_FOR\_ABIT,

APPROACH\_TOP\_LEVEL,

} SmartSearch\_State\_t;

static const char \*StateNames[] = {

"INIT\_STATE",

"WAITING",

"AVOID\_CENTER\_TAPE",

"AVOID\_LEFT\_TAPE",

"AVOID\_RIGHT\_TAPE",

"SWEEP\_LEFT",

"SWEEP\_RIGHT",

"RUN\_FOR\_ABIT",

"APPROACH\_TOP\_LEVEL",

};

#define APPROACH\_SPEED 600

#define TIME\_AVOID\_TAPE\_SHORT 700

#define TIME\_AVOID\_TAPE\_LONG 2500

#define TIME\_RUN\_A\_BIT 2000

#define SWEEP\_TIME\_LIMIT 900

#define AVOID\_SPEED\_SLOW 400

#define AVOID\_SPEED\_FAST 800

#define RUN\_SPEED 700

#define STOP\_SPEED 0

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

\* PRIVATE FUNCTION PROTOTYPES \*

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

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

\* PRIVATE MODULE VARIABLES \*

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

static SmartSearch\_State\_t CurrentState = INIT\_STATE; //Initialize static Variables

//static ES\_Event ThisEvent, returnEvent, InitEvent;

static uint8\_t MyPriority;

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

\* PUBLIC FUNCTIONS \*

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

/\*\*

\* @Function InitSmartSearch\_SubHSM(void)

\* @param Priority - internal variable to track which event queue to use

\* @return TRUE or FALSE

\* @brief This will get called by the framework at the beginning of the code

\* execution. It will post an ES\_INIT event to the appropriate event

\* queue, which will be handled inside RunSmartSearch\_SubHSM function.

\* Returns TRUE if successful, FALSE otherwise

\* @author J. Edward Carryer, 2011.10.23 19:25

\* @editor Brian Naranjo, 11/15/19 \*/

uint8\_t InitSmartSearch\_SubHSM(void) {

ES\_Event returnEvent;

CurrentState = INIT\_STATE;

// InitEvent.EventType = ES\_INIT;

returnEvent = RunSmartSearch\_SubHSM(INIT\_EVENT);

if (returnEvent.EventType == ES\_NO\_EVENT) {

return TRUE;

} else {

return FALSE;

}

}

/\*\*

\* @Function PostSmartSearch\_SubHSM(ES\_Event ThisEvent)

\* @param ThisEvent - the event (type and param) to be posted to queue

\* @return TRUE or FALSE

\* @brief This function is a wrapper to the queue posting function, and its name

\* will be used inside ES\_Configure to point to which queue events should

\* be posted to. Remember to rename to something appropriate.

\* Returns TRUE if successful, FALSE otherwise

\* @author J. Edward Carryer, 2011.10.23 19:25

\* @editor Brian Naranjo, 11/15/19 \*/

//uint8\_t PostSmartSearch\_SubHSM(ES\_Event ThisEvent) {

// return ES\_PostToService(MyPriority, ThisEvent);

//}

/\*\*

\* @Function RunSmartSearch\_SubHSM(ES\_Event ThisEvent)

\* @param ThisEvent - the event (type and param) to be responded.

\* @return Event - return event (type and param), in general should be ES\_NO\_EVENT

\* @brief This function is where you implement the whole of the service,

\* as this is called any time a new event is passed to the event queue.

\* @note Returns ES\_NO\_EVENT if the event have been "consumed."

\* @author J. Edward Carryer, 2011.10.23 19:25

\* @editor Brian Naranjo, 11/15/19 \*/

ES\_Event RunSmartSearch\_SubHSM(ES\_Event ThisEvent) {

uint8\_t makeTransition = FALSE;

SmartSearch\_State\_t NextState; //declare NextState Variable

ES\_Tattle(); // trace call stack

switch (CurrentState) {

case INIT\_STATE:

if (ThisEvent.EventType == ES\_INIT) {// only respond to ES\_Init

NextState = WAITING; //Transition to First State

makeTransition = TRUE;

ThisEvent.EventType = ES\_NO\_EVENT; //Initialize Event

}

break;

case WAITING:

Motors\_LeftDrive(APPROACH\_SPEED); //Set Both Motors to drive FWD

Motors\_RightDrive(APPROACH\_SPEED);

Motors\_LeftDirection(MOTOR\_FORWARD);

Motors\_RightDirection(MOTOR\_FORWARD);

// if (ThisEvent.EventType == TAPE\_FC\_TRIPPED) {

// NextState = AVOID\_CENTER\_TAPE;

// makeTransition = TRUE;

// ThisEvent.EventType = ES\_NO\_EVENT; //Initialize Event

// ES\_Timer\_InitTimer(SMARTSEARCH\_TIMER, TIME\_AVOID\_TAPE\_LONG); //Time Start for Avoid Long

// } else if (ThisEvent.EventType == TAPE\_FL\_TRIPPED) {

if (ThisEvent.EventType == TAPE\_FL\_TRIPPED) {

NextState = AVOID\_LEFT\_TAPE;

makeTransition = TRUE;

ThisEvent.EventType = ES\_NO\_EVENT; //Initialize Event

ES\_Timer\_InitTimer(SMARTSEARCH\_TIMER, TIME\_AVOID\_TAPE\_SHORT); //Time Start for Avoid Short

} else if (ThisEvent.EventType == TAPE\_FR\_TRIPPED) {

NextState = AVOID\_RIGHT\_TAPE;

makeTransition = TRUE;

ThisEvent.EventType = ES\_NO\_EVENT; //Initialize Event

ES\_Timer\_InitTimer(SMARTSEARCH\_TIMER, TIME\_AVOID\_TAPE\_SHORT); //Time Start for Avoid Short

}

break;

// case AVOID\_CENTER\_TAPE:

// Motors\_TankTurn(AVOID\_SPEED\_FAST, TANK\_TURN\_LEFT); //Tank Turn 90 degrees

// if (ThisEvent.EventType == ES\_TIMEOUT) {

// NextState = RUN\_FOR\_ABIT;

// makeTransition = TRUE;

// ThisEvent.EventType = ES\_NO\_EVENT; //Initialize Event

// ES\_Timer\_InitTimer(SMARTSEARCH\_TIMER, TIME\_RUN\_A\_BIT); //Time Start for Running

// }

// break;

case AVOID\_LEFT\_TAPE:

// Motors\_LeftDrive(AVOID\_SPEED\_SLOW); //Set Both Motors to Turn Right

// Motors\_RightDrive(AVOID\_SPEED\_FAST);

Motors\_TankTurn(APPROACH\_SPEED,TANK\_TURN\_LEFT);

if (ThisEvent.EventType == ES\_TIMEOUT) {

NextState = WAITING;

makeTransition = TRUE;

ThisEvent.EventType = ES\_NO\_EVENT; //Initialize Event

ES\_Timer\_InitTimer(SMARTSEARCH\_TIMER, SWEEP\_TIME\_LIMIT); //Time Start for Sweep Limit

}

break;

case AVOID\_RIGHT\_TAPE:

// Motors\_LeftDrive(AVOID\_SPEED\_FAST); //Set Both Motors to Turn Left

// Motors\_RightDrive(AVOID\_SPEED\_SLOW);

Motors\_TankTurn(APPROACH\_SPEED, TANK\_TURN\_RIGHT);

if (ThisEvent.EventType == ES\_TIMEOUT) {

NextState = WAITING;

makeTransition = TRUE;

ThisEvent.EventType = ES\_NO\_EVENT; //Initialize Event

ES\_Timer\_InitTimer(SMARTSEARCH\_TIMER, SWEEP\_TIME\_LIMIT); //Time Start for Sweep Limit

}

break;

// case SWEEP\_LEFT:

// Motors\_TankTurn(AVOID\_SPEED\_FAST, TANK\_TURN\_LEFT); //Sweep 180 Degrees Left

// if (ThisEvent.EventType == ES\_TIMEOUT) {

// NextState = WAITING;

// makeTransition = TRUE;

// ThisEvent.EventType = ES\_NO\_EVENT; //Initialize Event

// }

// // if (ThisEvent.EventType == BEACON\_FOUND){

// // NextState = APPROACH\_TOP\_LEVEL;

// // makeTransition = TRUE;

// // ThisEvent.EventType = ES\_NO\_EVENT; //Initialize Event

// // }

//

// break;

// case SWEEP\_RIGHT:

// Motors\_TankTurn(AVOID\_SPEED\_FAST, TANK\_TURN\_RIGHT); //Sweep 180 Degrees Right

// if (ThisEvent.EventType == ES\_TIMEOUT) {

// NextState = WAITING;

// makeTransition = TRUE;

// ThisEvent.EventType = ES\_NO\_EVENT; //Initialize Event

// }

// // if (ThisEvent.EventType == BEACON\_FOUND){

// // NextState = APPROACH\_TOP\_LEVEL;

// // makeTransition = TRUE;

// // ThisEvent.EventType = ES\_NO\_EVENT; //Initialize Event

// // }

// break;

// case RUN\_FOR\_ABIT:

// Motors\_LeftDrive(RUN\_SPEED); //Set Both Motors to drive FWD a bit

// Motors\_RightDrive(RUN\_SPEED);

// if (ThisEvent.EventType == ES\_TIMEOUT) {

// NextState = SWEEP\_RIGHT;

// makeTransition = TRUE;

// ThisEvent.EventType = ES\_NO\_EVENT; //Initialize Event

// ES\_Timer\_InitTimer(SMARTSEARCH\_TIMER, SWEEP\_TIME\_LIMIT); //Time Start for Sweep Limit

// }

// break;

// case APPROACH\_TOP\_LEVEL:

// ThisEvent = RunBotHSM(ThisEvent);

// ThisEvent.EventType = BEACON\_FOUND;

// break;

default:

break;

}

if (makeTransition == TRUE) { // making a state transition, send EXIT and ENTRY

// recursively call the current state with an exit event

RunSmartSearch\_SubHSM(EXIT\_EVENT); // <- Exit SM

CurrentState = NextState; //Change States

RunSmartSearch\_SubHSM(ENTRY\_EVENT); // <- Enter SM

}

ES\_Tail(); // trace call stack end

return ThisEvent;

}

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

\* PRIVATE FUNCTIONs \*

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