**LEGO MINDSTORMS NXT PROJECT**

**Course CS428 Embedded Systems**

**Computer Science Department University Of Crete**

Thodoris Pontzouktzidis csd4336

Dimitris Vlachos csd4492

Alexandros Tevrentzidis csd4383

Abstract

Timeline

* Create startup welcome message ✅
* Design a tree like data structure for menu implementation ✅
* Implement menu Visuals & functionality ✅
* Implement Morse translation utility functions ✅
* Implement touch sensor Morse input ✅
* Implement sound Morse output ✅
* Implement Mic Morse receive ✅
* Implement Mic Morse Send/Execute commands ✅
* **Welcome message.**

when the brick powers on display a logo frame and the first letters of each teammate name accompanied by the corresponding morse beeps.

* **Tree like data structure for menu.**

The menu is designed as a tree-like data structure.

Each menu option is a node on the tree. A node is a struct nav\_node :

typedef struct nav\_node{

enum navigation\_states\_t state;

int num\_of\_children;

struct nav\_node \*\*children;

int text\_len;

UBYTE \*\*text;

struct nav\_node \*back;

}navigation\_node;

**state** is used in the main menu switch inside **firmware.c** to understand in

which node-state we currently are also acting as node id.

enum navigation\_states\_t{MAIN\_MENU = 0, SEND = 1, RECEIVE = 2, BACK = 3, EXIT = 4, WRITE\_TXT = 5, VIEW\_TXT = 6, SEND\_SOUND = 7, SEND\_LED = 8, RECEIVE\_MIC = 9, RECEIVE\_LIGHT = 10, NOP = 11, WRITING\_TXT = 12, VIEWING\_TXT = 13, SENDING\_LED = 14, SENDING\_SOUND = 15};

**num\_of\_children** is the number of children of the node.

**children** is an array of pointers to the current node’s children.

**text\_len** each node holds a the menu text that is going to be displayed so we need

the text length of later.

**text** is an array with the node’s menu options in ascii.

**back** some nodes have a back edge to their prev.

Each level of the tree represents each level of the menu options :

We hardcode Initialize the nodes of the menu inside **navigation.c**

**2. Implement menu Visuals & functionality.**

Now inside **firmware.c** we have the menu switch :

switch(current\_nav\_state){

case MAIN\_MENU:

break;

case NOP:

break;

case SEND:

case RECEIVE:

DisplayMenu(CLEAR, current\_node->text, current\_node->text\_len);

DisplaySelectWord(CLEAR, current\_node->text, current\_node->text\_len, index\_of\_children);

current\_node = current\_node->children[index\_of\_children];

index\_of\_children = 0;

DisplayMenu(SET, current\_node->text, current\_node->text\_len);

DisplaySelectWord(SET, current\_node->text, current\_node->text\_len, index\_of\_children);

current\_nav\_state = NOP;

break;

case BACK:

DisplayMenu(CLEAR, current\_node->text, current\_node->text\_len);

DisplaySelectWord(CLEAR, current\_node->text, current\_node->text\_len, index\_of\_children);

DisplayString(CLEAR, 0, 0, str\_buf);

//DisplayString(CLEAR, 0, 0, morse\_buf);

current\_node = current\_node->children[index\_of\_children];

current\_node = current\_node->back;

index\_of\_children = 0;

DisplayMenu(SET, current\_node->text, current\_node->text\_len);

DisplaySelectWord(SET, current\_node->text, current\_node->text\_len, index\_of\_children);

current\_nav\_state = NOP;

break;

case WRITE\_TXT:

DisplayMenu(CLEAR, current\_node->text, current\_node->text\_len);

DisplaySelectWord(CLEAR, current\_node->text, current\_node->text\_len, index\_of\_children);

current\_node = current\_node->children[index\_of\_children];

index\_of\_children = 0;

DisplayMenu(SET, current\_node->text, current\_node->text\_len);

DisplaySelectWord(SET, current\_node->text, current\_node->text\_len, index\_of\_children);

current\_nav\_state = WRITING\_TXT;

current\_node = current\_node->children[index\_of\_children];

released = FALSE;

pressed = TRUE;

first\_time = TRUE;

ms\_p = 0;

ms\_r = 0;

i = 0;

memset((UBYTE\*)morse\_buf, 0, MORSE\_BUF\_LEN);

memset((UBYTE\*)str\_buf, 0, STR\_BUF\_LEN);

break;

case WRITING\_TXT:

if(t == TRUE){

if(pressed == TRUE){

ms\_p = GetMs();

time\_released = ms\_p - ms\_r;

if(first\_time){

//Sleep(500);

time\_released = 0;

}

first\_time = FALSE;

pressed = FALSE;

released = TRUE;

if(time\_released >= 500 && time\_released < 1000){

morse\_buf[i++]='?';

}else if(time\_released >= 1000){

morse\_buf[i++]=' ';

}

}

}else if(t == FALSE){

if(released == TRUE){

ms\_r = GetMs();

time\_pressed = ms\_r - ms\_p;

released = FALSE;

pressed = TRUE;

if(time\_pressed < 200){

morse\_buf[i++]='.';

}else{

morse\_buf[i++]='\_';

}

}

}

break;

case VIEW\_TXT:

DisplayMenu(CLEAR, current\_node->text, current\_node->text\_len);

DisplaySelectWord(CLEAR, current\_node->text, current\_node->text\_len, index\_of\_children);

current\_node = current\_node->children[index\_of\_children];

index\_of\_children = 0;

Morse2String(morse\_buf, str\_buf);

DisplayString(SET, 0, 0, str\_buf);

//DisplayString(SET, 0, 0, morse\_buf);

current\_nav\_state = VIEWING\_TXT;

current\_node = current\_node->children[index\_of\_children];

break;

case VIEWING\_TXT: // DISPLAY STRING UNTIL ENTER IS PRESSED

break;

case SEND\_LED:

DisplayMenu(CLEAR, current\_node->text, current\_node->text\_len);

DisplaySelectWord(CLEAR, current\_node->text, current\_node->text\_len, index\_of\_children);

current\_node = current\_node->children[index\_of\_children];

index\_of\_children = 0;

DisplayMenu(SET, current\_node->text, current\_node->text\_len);

DisplaySelectWord(SET, current\_node->text, current\_node->text\_len, index\_of\_children);

current\_nav\_state = SENDING\_LED;

current\_node = current\_node->children[index\_of\_children];

break;

case SENDING\_LED:

Sleep(500);

for(i=0; morse\_buf[i]!='\0'; i++){

switch(morse\_buf[i]){

case '.':

LedSwitchOn(LIGHT\_S);

Sleep(DOT\_TIME);

LedSwitchOff(LIGHT\_S);

Sleep(DOT\_TIME);

break;

case '\_':

LedSwitchOn(LIGHT\_S);

Sleep(DASH\_TIME);

LedSwitchOff(LIGHT\_S);

Sleep(DOT\_TIME);

break;

case '?':

Sleep(NEW\_LETTER\_TIME);

break;

case ' ':

Sleep(NEW\_WORD\_TIME);

break;

default:

break;

}

}

current\_nav\_state = current\_node->children[index\_of\_children]->state;

break;

case EXIT:

DisplayNum(SET,20,0,33);

DisplayUpdateSync();

I2CCtrl(POWERDOWN);

break;

}

We also have a second switch that controls the button input , changes the current\_nav\_state and displays some triangles around the current text option of the menu .

switch(ButtonRead()){

case BUTTON\_LEFT: // MOVE TO THE PREV NAVIGATION OPTION

if(btn\_is\_pressed) // MAKE SURE THAT THE BUTTON IS

TRIGGERED ONLY ONCE AFTER PRESSED

break;

btn\_is\_pressed = TRUE;

DisplaySelectWord(CLEAR, current\_node->text, current\_node->text\_len, index\_of\_children); // CLEAR THE OLD ARROWS

if(index\_of\_children == 0)

index\_of\_children = current\_node->num\_of\_children - 1;

else

index\_of\_children--;

DisplaySelectWord(SET, current\_node->text, current\_node->text\_len, index\_of\_children); // SET NEW ARROWS

break;

case BUTTON\_RIGHT: // MOVE TO THE NEXT NAVIGATION OPTION

if(btn\_is\_pressed)

break;

btn\_is\_pressed = TRUE;

DisplaySelectWord(CLEAR, current\_node->text, current\_node->text\_len, index\_of\_children); // CLEAR THE OLD ARROWS

index\_of\_children ++;

index\_of\_children = index\_of\_children % current\_node->num\_of\_children;

DisplaySelectWord(SET, current\_node->text, current\_node->text\_len, index\_of\_children); // CLEAR THE OLD ARROWS

break;

case BUTTON\_ENTER:

if(btn\_is\_pressed)

break;

btn\_is\_pressed = TRUE;

current\_nav\_state = current\_node->children[index\_of\_children]->state;

break;

case BUTTON\_EXIT:

if(btn\_is\_pressed)

break;

btn\_is\_pressed = TRUE;

I2CCtrl(REPROGRAM);

break;

case BUTTON\_NONE:

btn\_is\_pressed = FALSE;

break;

default:

break;

}

Everything for the menu visuals and functionality happens in these 2 switches. We change the current\_nav\_state depending on which node-menu option we are and which button is pressed. We also have a variable index\_of\_children that acts as an index for changing the current node everytime we click enter and we select a menu option and as an index for selecting the correct text of the current node and outline it with arrows.