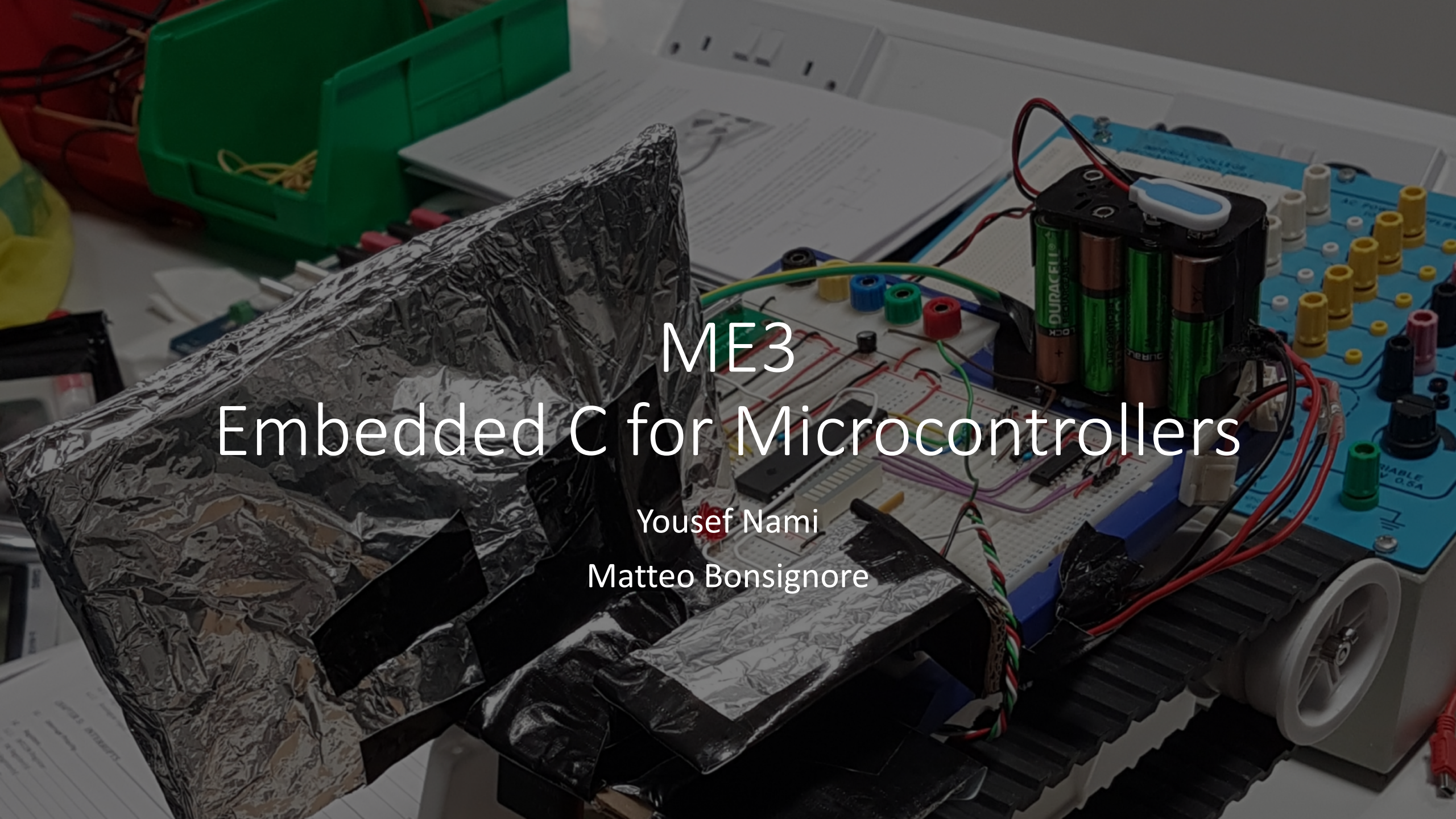


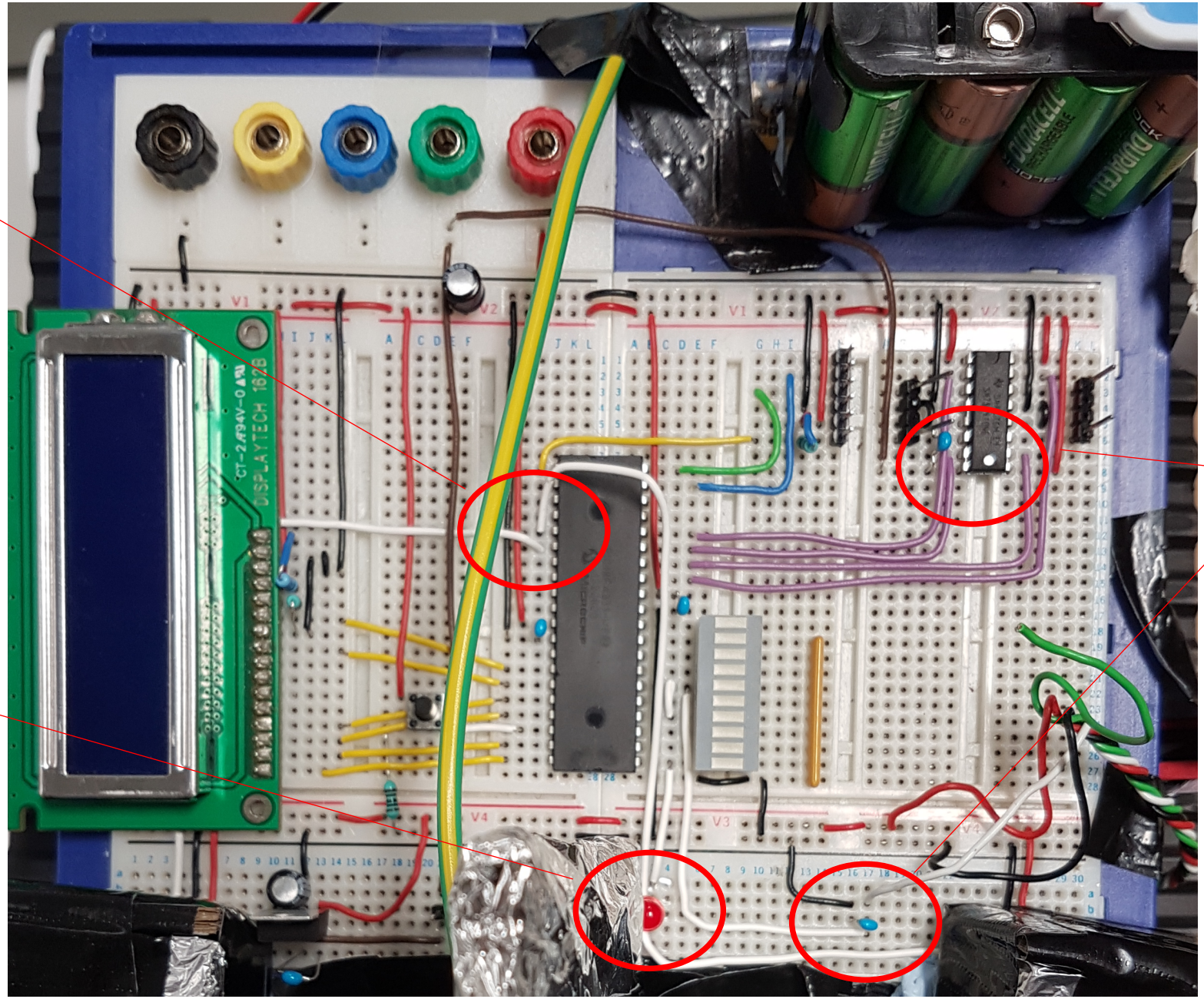
ME3
Embedded C for Microcontrollers

Yousef Nami

Matteo Bonsignore

CAP2/ CAP3 for IR sensors

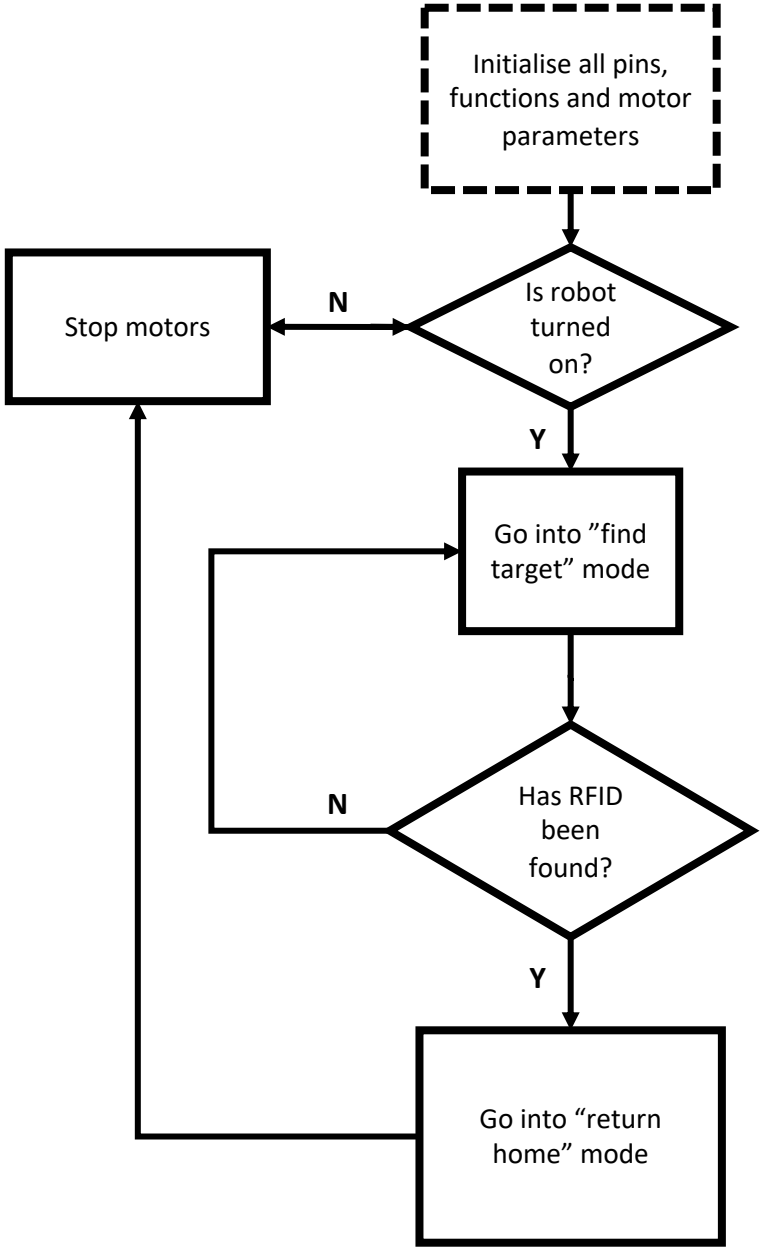
LED for ON/OFF visual aid



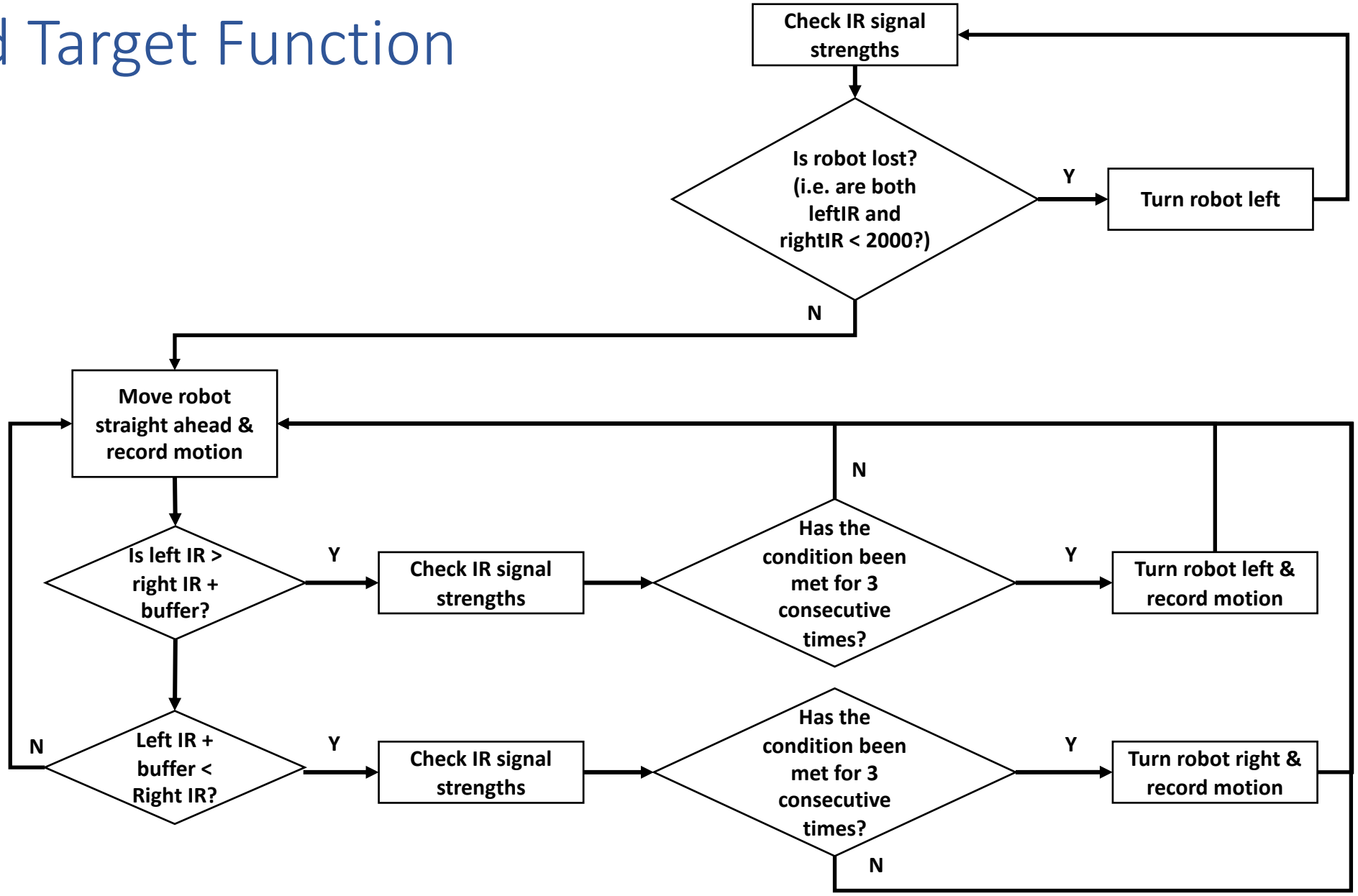
NOT PRESENTED

Capacitors for smoothing out ripples

Main Control Loop



Find Target Function



Motor Drive

```
94  //function to make the robot turn left
95  void turnLeft(struct DC_motor *mL, struct DC_motor *mR)
96  {
97          mL->direction=1;
98          mR->direction=0;
99          setMotorsFullSpeed(mL, mR);
100         stop(mL, mR);
101 }
102
103 //function to make the robot turn right
104 void turnRight(struct DC_motor *mL, struct DC_motor *mR)
105 {
106         mL->direction=0;
107         mR->direction=1;
108         setMotorsFullSpeed(mL, mR);
109         stop(mL, mR);
110 }
111
112 //function to make the robot go straight
113 void fullSpeedAhead(struct DC_motor *mL, struct DC_motor *mR)
114 {
115          mL->direction=0;
116         mR->direction=0;
117         setMotorsFullSpeed(mL, mR);
118         stop(mL, mR);
119 }
120
```

OLD CODE USED IN MOTOR LAB

Need function for backwards motion

Not very elegant

Motor Drive

```
132  /*function to move motor*/
133
134      void move(struct DC_motor *mL,struct DC_motor *mR, char dir){
135          mL->direction=(dir>>1)&(0b01);
136          mR->direction=(dir)&(0b01);
137          setMotorsFullSpeed(mL,mR);
138          stop(mL,mR);
139      }
```

NEW IMPLEMENTATION

GUIDE

0b00 => forwards

0b11 => back

0b10 => left

0b01 => right

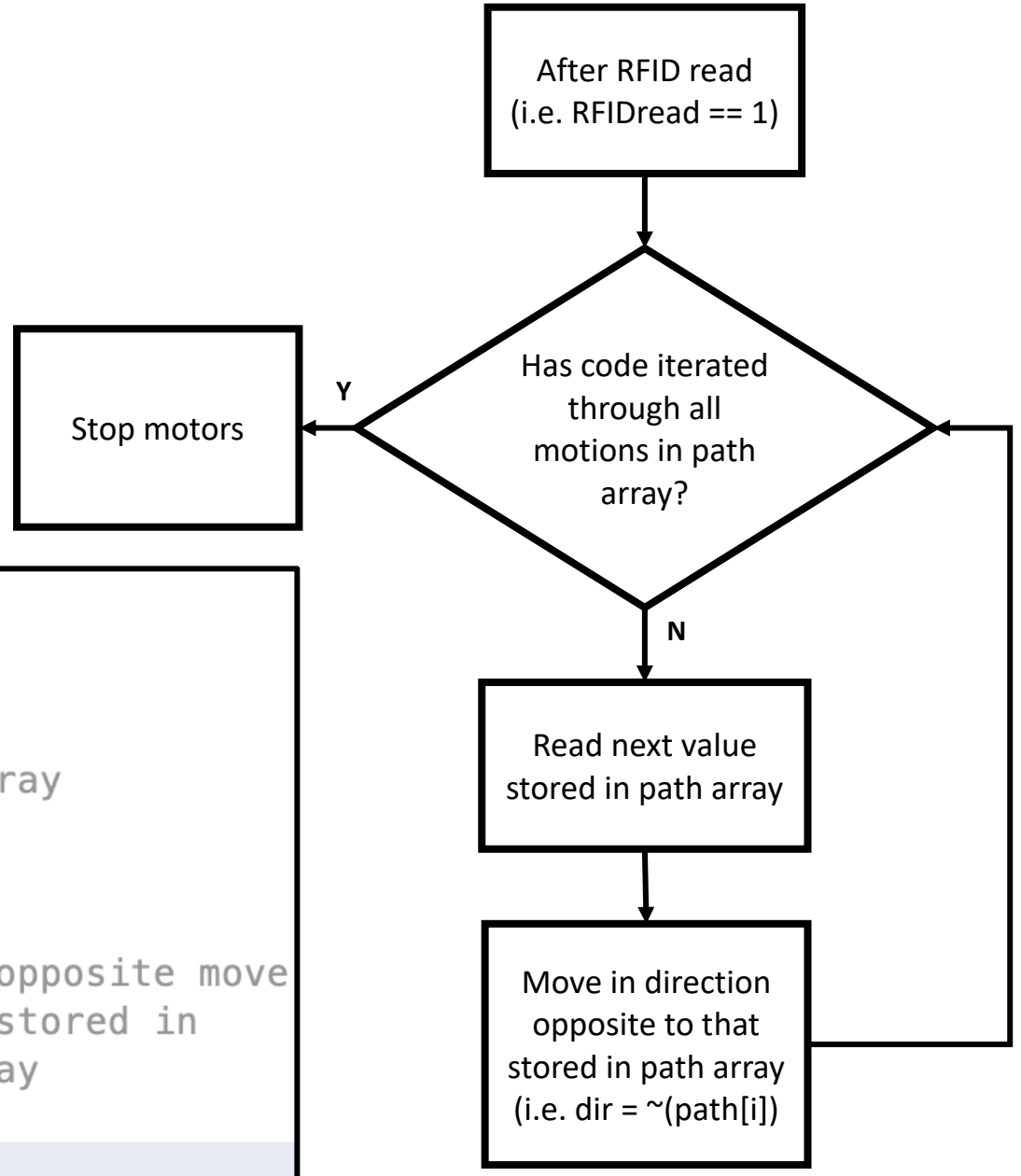
Motion functions modified for efficiency

Motion function now also takes “**char** dir” to determine direction

- ✓ Reduced number of motion functions (1 vs 3 in motor lab)
- ✓ More efficient implementation of return function (see next slide)

Return Home Function

- New motion function allows motions to be recorded in path array as directions according to guide
- This allows Return Home function to be implemented with a single while () loop, as shown below




```
if (RFIDread == 1) {

    while (i > 0) { //iterate backwards though path array

        --i;

        move(&motorL, &motorR, ~(path[i])); //perform opposite move
                                            //of that stored in
                                            //path array

    }
```

IR Sensor Readings

```
/* Read strength of IR signals as binary value */

    void findStrengths(int *leftIR, int *rightIR) { //receive pointers for
                                                    //left and right IR signals

    //record pulse widths of the left & right IR signals
    *leftIR = (CAP2BUFL | (CAP2BUFH<<8));
    *rightIR = (CAP3BUFL | (CAP3BUFH<<8));

}
```

The left and right IR sensors are connected to CAP2 and CAP3 pins (respectively) and provide a PWM signal

The CAP2 and CAP3 bits are set to pulse-width measurement mode using Timer 5 and are set to measure signals from falling to rising edge (which is proportional to the respective IR signal strength)

The values of the IR signal strengths (as measured by Timer 5) are written to CAPxBUFL & CAPxBUFLH, which are combined as above and stored in two different variables (int leftIR and int rightIR)

For efficiency, the **findStrengths** function takes pointers to int leftIR and int rightIR (defined in main() function) and updates their values directly from the function, avoiding the need for the function to return a value

Obtain RFID Code

Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(	72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29	)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-_	63	3F	?	95	5F	_	127	7F	DEL

```
20              a = getCharSerial();
21              if (index <= 12) {
22                  SendLCD(a, 1);
23                  //converts ASCII characters to their decimal values
24                  if (a > 64) { //if ASCII character is a letter
25                      a = a - 55;
26                  } else { //if it's a number
27                      a = a - 48;
28                  }
29
30                  currentVal = (a | currentVal);
31
32                  if (index % 2 == 1) { //if its odd
33                      currentVal = currentVal << 4;
34                  } else { //if its even
35                      if (index < 11) {
36                          checkSum = checkSum^currentVal;
37                      } else {
38                          if (checkSum == currentVal) {
39                              SetLine(2);
40                              LCD_String("Match");
41                          } else {
42                              SetLine(2);
43                              LCD_String("No Match");
44                          }
45                      }
46                      currentVal = 0;
47                  }
48                  index = index + 1;
49              }
```

Set Motor PWM Function Efficiency

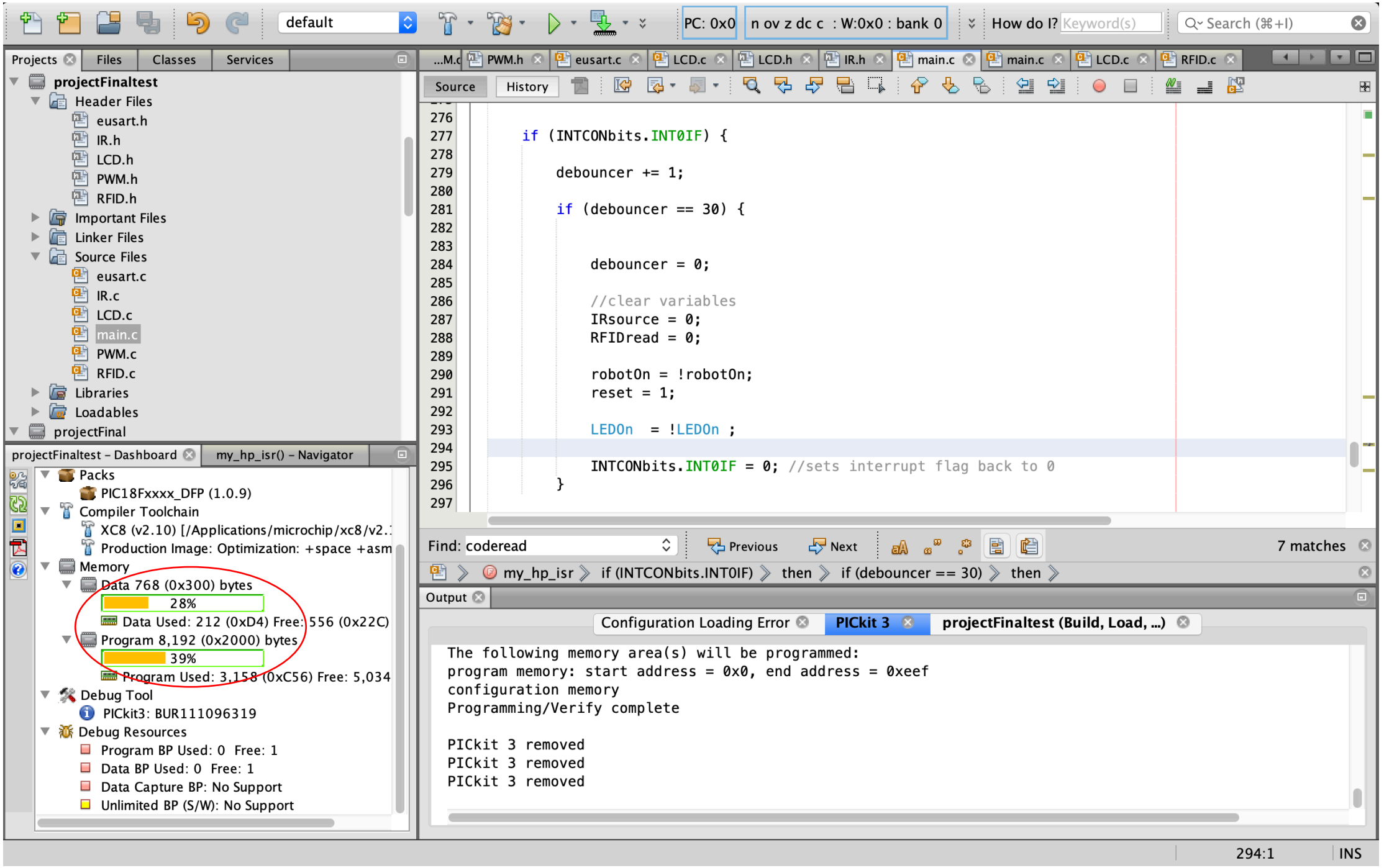
```
81  void setMotorPWM(struct DC_motor *m){
82
83      int PWMduty;
84      if (m->direction) {
85          PWMduty=m->PWMperiod-((int)m->power)*(m->PWMperiod)/100;
86      }
87      else {
88          PWMduty=((int)m->power)*(m->PWMperiod)/100;
89      }
90
91      PWMduty=(PWMduty<<2);
92      *(m->dutyLowByte)=PWMduty&0xFF;
93      *(m->dutyHighByte)=(PWMduty>>8)&0x3F;
94
95      if (m->direction) {
96          LATB=LATB | (1<<(m->dir_pin));
97     } else {
98          LATB=LATB&(~(1<<(m->dir_pin)));
99      }
100
101 }
```

```
81  void setMotorPWM(struct DC_motor *m){
82
83      int PWMduty = ((int)m->power)*(m->PWMperiod);
84      if (m->direction) {
85          PWMduty = (PWMduty>>9)+PWMduty>>7;
86          PWMduty = m->PWMperiod - PWMduty;
87      }
88      else {
89          PWMduty = (PWMduty>>9)+PWMduty>>7;
90      }
91
92      PWMduty=(PWMduty<<2);
93      *(m->dutyLowByte)=PWMduty&0xFF;
94      *(m->dutyHighByte)=(PWMduty>>8)&0x3F;
95
96      if (m->direction) {
97          LATB=LATB | (1<<(m->dir_pin));
98     } else {
99          LATB=LATB&(~(1<<(m->dir_pin)));
100     }
101
102 }
```

Example:

$$\frac{x}{100} \approx \frac{x}{2^9} + \frac{x}{2^7}$$

$$\varepsilon \approx -2\%$$


default
PC: 0x0
n ov z dc c : W:0x0 : bank 0
How do I? Keyword(s)
Search (⌘+I)
Projects
Files
Classes
Services
projectFinaltest
Header Files
eusart.h
IR.h
LCD.h
PWM.h
RFID.h
Important Files
Linker Files
Source Files
eusart.c
IR.c
LCD.c
main.c
PWM.c
RFID.c
Libraries
Loadables
projectFinal
projectFinaltest – Dashboard
my_hp_isr() – Navigator
Packs
PIC18Fxxxx_DFP (1.0.9)
Compiler Toolchain
XC8 (v2.10) [/Applications/microchip/xc8/v2.
Production Image: Optimization: +space +asm
Memory
Data 768 (0x300) bytes
28%
Data Used: 212 (0xD4) Free: 556 (0x22C)
Program 8,192 (0x2000) bytes
39%
Program Used: 3,158 (0xC56) Free: 5,034
Debug Tool
PICkit3: BUR111096319
Debug Resources
Program BP Used: 0 Free: 1
Data BP Used: 0 Free: 1
Data Capture BP: No Support
Unlimited BP (S/W): No Support
...M.c
PWM.h
eusart.c
LCD.c
LCD.h
IR.h
main.c
main.c
LCD.c
RFID.c
Source
History
276
277 if (INTCONbits.INT0IF) {
278
279 debouncer += 1;
280
281 if (debouncer == 30) {
282
283
284 debouncer = 0;
285
286 //clear variables
287 IRsource = 0;
288 RFIDread = 0;
289
290 robotOn = !robotOn;
291 reset = 1;
292
293 LEDOn = !LEDOn ;
294
295 INTCONbits.INT0IF = 0; //sets interrupt flag back to 0
296 }
297
Find: coderead
Previous
Next
7 matches
my_hp_isr
if (INTCONbits.INT0IF)
then
if (debouncer == 30)
then
Output
Configuration Loading Error
PICkit 3
projectFinaltest (Build, Load, ...)
The following memory area(s) will be programmed:
program memory: start address = 0x0, end address = 0xeef
configuration memory
Programming/Verify complete
PICkit 3 removed
PICkit 3 removed
PICkit 3 removed
294:1
INS

Further Improvements

NOT PRESENTED

- The difference in IR signal strength could be measured to gauge the angle the car has to turn by at every step
- Provided the memory is sufficient, multiple readings could be taken from the IR sensors at each stage and the median value used to determine in which direction the robot should turn
- The LDR could be configured to change the behavior of the car depending on the ambient light level