EE4930 Advanced Microprocessor

Section 011/021, Winter 2022/23

Professor: Kerry R. Widder, Ph.D.

Electrical Engineering and Computer Science Department

Milwaukee School of Engineering

Final Exam : “Igloo”

Name: Jorge Jurado-Garcia

Courses: EE 4930 011

Date: 02/17/2023

# Objectives

The purpose of this final exam was to expand our implementation of the MSP432 bord to control the heating capabilities of an igloo. We were given the possibility of utilizing TI-RTOS to implement the ADC, PWM, and Timer to take readings from a pushbutton or potentiometer which converts the reading to a temperature setpoint value between 50-90 F. While this is running the temperature sensor reads the temperature inside our igloo and decides where to turn on the fan for heating or off. A hysteresis of +1/-1 was given for the temp.

# Description

For this final exam I reused most of code from previous labs. After testing functionality with generic potentiometers, LEDs, and switches I verified that my FSM machine was operating correctly. Next, I went on to attempt to build the actual circuit but ran into some issues with setting up the transistors for the Fan and Heater. This took most of my time since I hard some faulty components that I had to debug and replace and modify some my code due to some gpio pins being unusable from overcurrent being suppled. Once I got connections successfully established and checked everything was working with the circuit I went ahead and implemented the correct conversion formula and gave it one more functional testing.

# Conclusion

For this final exam I decided to use the LUT and used my week 4 lab as a starting point for this final exam. I reused most of my code from previous labs and manly only modified some values, LUT, and the read event function for the code to work properly. Most of my problems arose from building the circuit and debugging since I ended up frying one of my pins on the board for reading the A/D and did not have all the needed components for the circuit. But once I have gotten all the circuit connections in place and build my paper igloo it was dandy. The main difficulties I encountered were solving hardware connection problems, which caused many issues. I think this lab serves a good final exam as it encompasses everything, we have learned in this quarter so far and was not to challenging to complete without an RTOS .Attachments

## Schematic

Diagram

Description automatically generated

## Main

/\* Copyright (C) 2022 MSOE

\*

\* All Rights Reserved

\* You may not use, distribute or modify this code without the

\* express written permission of MSOE

\*

\* Contact Info

\* jorgejuradogarcia2@gmail.com

\* 608-312-5950

\*

\*/

**#include** <stdio.h>

**#include** "msp432.h"

**#include** "msoe\_lib\_all.h"

**#include** "defines.h"

**#include** <MSP432P401R\_GPIO.h>

//Preprocessors

**#define** LAB1\_ON FALSE

**#define** LAB2\_ON FALSE

**#define** LAB3\_ON FALSE

**#define** LAB4\_ON FALSE

**#define** LAB5\_ON FALSE

**#define** LAB6\_ON FALSE

**#define** FINAL\_EXAM TRUE

// Lab modules

**#if** FINAL\_EXAM

**#include** "EE4930\_FINAL\_EXAM.h"

eSystemInputs Inputs;

**#endif**

/\*

\* main.c

\*/

**int** **main**(**void**){

**#if** FINAL\_EXAM

Igloo\_init();

eSystemEvent eNewEvent = Last\_event;

eSystemState eNextState = Off\_state;

**while**(1)

{

eNewEvent = Igloo\_ReadEvent();

//poll of the state machine

eNextState = Igloo\_Poll( eNewEvent, eNextState );

Igloo\_UpdateLCD();

}

**#endif**

} // end main

## EE4930\_FINALEXAM.c

/\*

\* EE4930\_FINAL\_EXAM.c

\*

\* Created on: Feb 14, 2023

\* Author: jurado-garciaj

\*/

**#include** "msp432.h"

**#include** "msoe\_lib\_all.h"

**#include** "EE4930\_FINAL\_EXAM.h"

**#include** "defines.h"

// Drivers

**#include** <MSP432P401R\_GPIO.h>

SIgloo Igloo;

**static** **float** temp = 0;

//file scope functions

**static** **void** **setup\_inputs**();

**static** **void** **setup\_outputs**();

**static** eSystemState **Temp\_above\_setpoint\_Handler**(**void**);

**static** eSystemState **Temp\_below\_setpoint\_Handler**(**void**);

**static** sStateMachine sMachiney [LOOPUP\_ROW][LOOPUP\_COLUMN] =

{

{

//off state moves

{Off\_state,*Temp\_above\_setpoint\_event*,Temp\_above\_setpoint\_Handler},

{Off\_state,*Temp\_below\_setpoint\_event*,Temp\_below\_setpoint\_Handler},

},

{

//on state moves

{Active\_state,*Temp\_above\_setpoint\_event*,Temp\_above\_setpoint\_Handler},

{Active\_state,*Temp\_below\_setpoint\_event*,Temp\_below\_setpoint\_Handler},

},

};

**void** **Igloo\_init**( **void** )

{

Igloo.sInputs.Igloo\_temp = 0;

Igloo.sInputs.setpoint\_temp = 0;

Igloo.sOutputs.Fan\_control = OFF;

Igloo.sOutputs.Heating\_Coil = OFF;

Igloo.sOutputs.LCD\_Display\_flag = OFF;

//init the gpio pins inputs

setup\_inputs();

//init the gpio pins outputs

setup\_outputs();

//MCU clock init

//Clock\_Init\_48MHz();

//Interrupt setup

NVIC->ISER[0] |= ( 1 << 24 ); //NVIC for ADC14 at ISER[24]

NVIC->ISER[1] |= ( 1 << 3 ); //NVIC for PORT 1 at ISER[35]

NVIC->ISER[0] |= ( 1 << 25 ); //NVIC for TIMER\_32\_1 at

\_\_enable\_interrupt();

**return**;

}

//Read event

eSystemEvent **Igloo\_ReadEvent**( **void** )

{

eSystemEvent return\_event = *no\_new\_event*;

//check if the sensor readings are below or above setpoints

**if**( (Igloo.sInputs.Igloo\_temp ) > (Igloo.sInputs.setpoint\_temp) )

{

//turn off

return\_event = *Temp\_above\_setpoint\_event*;

}

**if**( (Igloo.sInputs.Igloo\_temp ) < (Igloo.sInputs.setpoint\_temp) )

{

//turn on

return\_event = *Temp\_below\_setpoint\_event*;

}

//return no handler

**return** return\_event;

}

//polling of the Igloo LUK

eSystemState **Igloo\_Poll**( eSystemEvent NewEvent, eSystemState NextState)

{

//Check NULL pointer and array boundary

**if**( ( NextState < last\_state) && (NewEvent < Last\_event) )

{

**int** i;

//loop around the state machine until the state was found

**for**( i=0; i < LOOPUP\_COLUMN; i ++ )

{

//if the Event machines and Stata machine Event Handel also matches

**if**( (sMachiney[NextState][i].eStateMachineEvent == NewEvent ) && (sMachiney[NextState][i].pfStateMachineEvnentHandler != NULL ) )

{

//function call for the next new state

// function call as per the state and event and return the next state of the finite state machine

NextState = (\*sMachiney[NextState][i].pfStateMachineEvnentHandler)();

}

} //end of for loops

}

**return** NextState;

}

**void** **Igloo\_UpdateLCD**( **void** )

{

**if**( Igloo.sOutputs.LCD\_Display\_flag)

{

//update the lcd

//update the temp and setpoint

LCD\_goto\_xy(7,1);

LCD\_print\_str(" ");

LCD\_goto\_xy(7,1);

LCD\_print\_dec3(Igloo.sInputs.Igloo\_temp);

LCD\_goto\_xy(7,2);

LCD\_print\_str(" ");

LCD\_goto\_xy(7,2);

LCD\_print\_dec3(Igloo.sInputs.setpoint\_temp);

Igloo.sOutputs.LCD\_Display\_flag = FALSE;

}

**return**;

}

//handler above set point

**static** eSystemState **Temp\_above\_setpoint\_Handler**( **void** )

{

//turn off fan

//GPIO\_setOutputLowOnPin( GPIO\_PORT\_P6, GPIO\_PIN6 );

//TIMER\_A2->CCR[0] = 0;

Config\_Duty(0);

//turn off coil

GPIO\_setOutputLowOnPin( GPIO\_PORT\_P4, GPIO\_PIN2 );

Igloo.sOutputs.Fan\_control = OFF;

Igloo.sOutputs.Heating\_Coil = OFF;

LCD\_goto\_xy(0,3);

LCD\_print\_str("FAN: OFF");

LCD\_goto\_xy(0,4);

LCD\_print\_str("Coil: OFF");

**return** Off\_state;

}

//handler below set point

**static** eSystemState **Temp\_below\_setpoint\_Handler**( **void** )

{

//turn on coils

GPIO\_setOutputHighOnPin( GPIO\_PORT\_P4, GPIO\_PIN2 );

//turn on fan

GPIO\_setOutputHighOnPin( GPIO\_PORT\_P6, GPIO\_PIN6 );

Config\_Duty(20);

Igloo.sOutputs.Fan\_control = ON;

Igloo.sOutputs.Heating\_Coil = ON;

LCD\_goto\_xy(0,3);

LCD\_print\_str("FAN: ON ");

LCD\_goto\_xy(0,4);

LCD\_print\_str("Coil: ON ");

**return** Active\_state;

}

//Init the setup\_inits

**void** **setup\_inputs**( **void** )

{

Stop\_watchdog(); // stop Watch dog timer

Set\_ports\_to\_out();

// For Humidity Setpoint - two pushbutton switches

// SW1

GPIO\_setAsInputPinWithPullUpResistor( GPIO\_PORT\_P1, GPIO\_PIN1 );

//Set P1.1 as an Input with Pull up resistor high

P1->IES &= ~GPIO\_PIN1; // enable edge select

P1->IE |= GPIO\_PIN1; // enable interrupt

// SW2

GPIO\_setAsInputPinWithPullUpResistor( GPIO\_PORT\_P1, GPIO\_PIN4 );

//Set P1.4 as an Input with Pull up resistor high

P1->IES &= ~GPIO\_PIN4; // enable edge select

P1->IE |= GPIO\_PIN4; // enable interrupt

// Potentiometer for Igloo Temperature

//A6

// input on A3 into MEM1

P5->SEL0 |= BIT2; // use with A3

P5->SEL1 |= BIT2; // use with A3

// Setting up the control register

ADC14->CTL0 = ADC14\_CTL0\_SHT0\_5 | ADC14\_CTL0\_SHP |

ADC14\_CTL0\_SSEL\_\_SMCLK | ADC14\_CTL0\_ON | ADC14\_CTL0\_CONSEQ\_1;

// Sampling time, S&H=96, ADC14 on, SMCLK, with 14 Bit Resolutions

// make sure read the reset operation first

// memory registers is used for a single conversion

//or for the first conversion in a sequence.

// setting the memory to MEM1

ADC14->CTL1 &= ~ADC14\_CTL1\_RES\_1;

ADC14->CTL1 |= (1 << ADC14\_CTL1\_CSTARTADD\_OFS) | ADC14\_CTL1\_RES\_2;

//Memory conversion control 0 register being set to get data for A3

ADC14->MCTL[1] |= ( ADC14\_MCTLN\_INCH\_3 | ADC14\_MCTLN\_EOS );

// Enabled interrupt information for mem[1]

ADC14->IER0 |= (1<<1);

// enable the conversion

ADC14->CTL0 |= ADC14\_CTL0\_ENC;

//setting up of the timer32

// Set master clock (MCLK) to HFXTCLK with divide by 1 - 48MHz

TIMER32\_1->LOAD |= 26875;

/\*

\* TIMER CLOCK SPEED: 48MHz MHz

\* TImer disabled

\* Periodic mode Enabled

\* Timer Interrupt enabled

\* Clock division of 256

\* Clock After Division in 48MHz/256 = 0.1875 MHz = 187,500 Hz

\* Size of 16-bit counter

\* Ticks 16 bits = 65535 ticks

\* Timer\_sec = 1/187500 = 5.333\*10^(-6)

\* Timer After expiration = 5.333\*10^(-6) \* 46875 = 0.25

\* Wrapping mode

\* 32-bit counter

\*/

/\*

\*

\*/

TIMER32\_1->CONTROL |= 0x6A;

// ENABLE TIMER

TIMER32\_1->CONTROL |= 0x80;

**return**;

}

//Init the setup\_outputs

**void** **setup\_outputs**( **void** )

{

//Fan control Output control turn on 5V DC fan 6.6

//setup of this gpio pin to a pwm signal of 17.5 Hz and 20% duty cycle

//PWM PINS

GPIO\_setAsOutputPin( GPIO\_PORT\_P6, GPIO\_PIN6); // P6.6 as an output pin

GPIO\_setOutputLowOnPin( GPIO\_PORT\_P6, GPIO\_PIN6 );

// input on A0 into MEM0

P6->SEL0 |= GPIO\_PIN6; // use with TA0.2

P6->SEL1 &= ~GPIO\_PIN6; // use with TA0.2

// SET the TIMER in UP MODE WITH RESER/SET MODE

// CLOCK SOURCE SMCLK 3MHZ

// DIVIDE by 8

// Mode Control: Up Mode

// Interrupt for right now disabled

TIMER\_A2->CTL |= (TIMER\_A\_CTL\_SSEL\_\_SMCLK|TIMER\_A\_CTL\_ID\_3|TIMER\_A\_CTL\_MC\_\_UP);

//TA0.2

// Set output mode to Reset and SET

TIMER\_A2->CCTL[3] |= (TIMER\_A\_CCTLN\_OUTMOD\_7);

// Capture Control register to TIMER\_Ax Capture.Compare

//12 MHZ /3 = 4,000,000 input frequency

// T\_timer = 1/f\_systemclock = 1/4 MHz = 0.25 uS

// CTL Frequency = 100K HZ frequency

// If using the full 16-bit timer we have 65535 ticks

// For maximum speed before tick overflow we will have

// t\_overflow = tick\_max \* T\_timer = 65535 \* 0.25 us = 16.38375 mS

//This is fast be good enough for our info

TIMER\_A2->CCR[0] = 65534;

//change this file in the Capture control register to change duty cycle

//TIMER\_A2->CCR[3] = 3250;

//setup of the gpio pin of the heater coil 4.2

GPIO\_setAsOutputPin( GPIO\_PORT\_P4, GPIO\_PIN2 );

GPIO\_setOutputLowOnPin( GPIO\_PORT\_P4, GPIO\_PIN2 );

// Configures the LCD display screen clears display afterwards

LCD\_Config();

//Printing "EE4930-011" on the first line of the LCD screen

LCD\_goto\_xy(0,1);

LCD\_print\_str("Temp F:");

LCD\_goto\_xy(0,2);

LCD\_print\_str("Setpt:");

LCD\_goto\_xy(0,3);

LCD\_print\_str("FAN: ");

LCD\_goto\_xy(0,4);

LCD\_print\_str("Coil: ");

**return**;

}

//setting up of the duty cycle for pwm

**int** **Config\_Duty**(**float** duty)

{

//Duty 0-100

//will need to grab current CCR[0] register value

//and also grab the its current duty cycle also

//current CCR[0] value:

**int** current\_val = TIMER\_A2->CCR[0];

//which direction

//pwmA which is P2.6 or PWMA

**float** new\_val = (**float**)( current\_val\*(duty/100) );

TIMER\_A2->CCR[3] = (**int**) new\_val;

**return** (**float**)(100\*new\_val)/current\_val;

}

// Interrupt Handler for SW1

**void** **PORT1\_IRQHandler**()

{

**int** readIV = P1->IV; // Reading IV register to clear

**switch**( readIV )

{

**case** 0x04: // P1.1

Igloo.sInputs.setpoint\_temp += 1;

Igloo.sOutputs.LCD\_Display\_flag = TRUE;

**break**;

**case** 0x0A: // P1.4

Igloo.sInputs.setpoint\_temp -= 1;

Igloo.sOutputs.LCD\_Display\_flag = TRUE;

**break**;

}

**if**(Igloo.sInputs.setpoint\_temp > 250)

{

Igloo.sInputs.setpoint\_temp = -250;

}

**if**( Igloo.sInputs.setpoint\_temp < -250 )

{

Igloo.sInputs.setpoint\_temp = 250;

}

}

// Interrupt Handler for TIMER32

**void** **T32\_INT1\_IRQHandler**()

{

TIMER32\_1->INTCLR = CLEAR;

// start a new A/D conversion

ADC14->CTL0 |= ADC14\_CTL0\_SC;

}

//Interrupt Handler for ADC14

**void** **ADC14\_IRQHandler**(**void**)

{

**static** **int** previous\_adc = 0;

**int** readIV = ADC14->IV; // Reading the IV register to clear

**int** adc\_reading = ADC14->MEM[1];

**float** adc\_voltage = ( ( (**float**)(adc\_reading)/4096 ) \* (3300.0f));

**int** temp\_c = adc\_voltage/10 - 50; // y = x10mV/C - 75

temp = ( temp\_c \* (9.0f / 5.0f) ) + 32.0;

Igloo.sInputs.Igloo\_temp = (**int**)(temp);

//check if the value is different

**if**( previous\_adc != Igloo.sInputs.Igloo\_temp )

{

Igloo.sOutputs.LCD\_Display\_flag = TRUE;

}

previous\_adc = Igloo.sInputs.Igloo\_temp;

}

## EE4930\_ FINALEXAM.h

/\*

\* EE4930\_FINAL\_EXAM.h

\*

\* Created on: Feb 14, 2023

\* Author: jurado-garciaj

\*/

**#ifndef** APP\_EE4930\_FINAL\_EXAM\_H\_

**#define** APP\_EE4930\_FINAL\_EXAM\_H\_

**#include** "defines.h"

//defs

**#define** LOOPUP\_COLUMN 2

**#define** LOOPUP\_ROW 2

//Different states

**typedef** **enum**

{

*Off\_state*,

*Active\_state*,

*last\_state*

} eSystemState;

//Different type events

**typedef** **enum**

{

*Temp\_above\_setpoint\_event*,

*Temp\_below\_setpoint\_event*,

*no\_new\_event*,

*Last\_event*

} eSystemEvent;

//input structure of state machine

**typedef** **struct**

{

**signed** **short** Igloo\_temp;

**signed** **short** setpoint\_temp;

} eSystemInputs;

//output structure of state machine

**typedef** **struct**

{

**char** Fan\_control;

**char** Heating\_Coil;

**char** LCD\_Display\_flag;

} eSystemOutputs;

//typedef of function pointer

**typedef** eSystemState (\*pfEventHandler)(**void**);

//structure of state and event with event handler

**typedef** **struct**

{

eSystemState eStateMachine;

eSystemEvent eStateMachineEvent;

pfEventHandler pfStateMachineEvnentHandler;

} sStateMachine;

//structure of module itself

**typedef** **struct**

{

eSystemInputs sInputs;

eSystemOutputs sOutputs;

} SIgloo;

**void** **Igloo\_init**( **void** );

**int** **Config\_Duty**( **float** duty );

eSystemEvent **Igloo\_ReadEvent**( **void** );

eSystemState **Igloo\_Poll**( eSystemEvent NewEvent, eSystemState NextState);

**void** **Igloo\_UpdateLCD**( **void** );

**#endif** /\* APP\_EE4930\_FINAL\_EXAM\_H\_ \*/