THE UNIVERSITY OF HUDDERSFIELD

School of Computing and Engineering

ASSIGNMENT SPECIFICATION

Module details	
Module Code	NIE2206
Module Title	Embedded Systems
Course Title/s	BEng/MEng Programmes (Electrical)

Assessment weighting, type and contact details		
Title	Project Work	
Weighting	26%	
Mode of working for assessment task	Individual Note: if the assessment task is to be completed on an individual basis there should be no collusion or collaboration whilst working on and subsequently submitting this assignment.	
Module Leader	Dr Haydn Martin	Contact details: h.p.martin@hud.ac.uk
Module Tutor/s		

Submission and feedback details	
Hand-out date	14/3/22
How to submit your work.	You must Brightspace
Submission date/s and times	16:00 on 2/5/22
Expected amount of independent time you should allocate to complete this assessment	10 hours
Submission type and format	Project code is to be submitted electronically to Unilearn by the specified deadline. The **whole project directory** must be submitted as a .zip file and **must build with no errors**. If the submitted project files do not build, you will score zero for the code functionality criterion.
Date by which your grade and feedback will be returned	23/5/22

Additional guidance information	
Your responsibility	It is your responsibility to read and understand the University regulations regarding conduct in assessment. Please pay special attention to the assessment regulations (section 4) on Academic Misconduct. In brief: ensure that you; 1. DO NOT use the work of another student - this includes students from previous years and other institutions, as well as current students on the module. 2. DO NOT make your work available or leave insecure, for other students to view or use. 3. Any examples provided by the module tutor should be appropriately referenced, as should examples from external sources. Further guidance can be found in the SCEN Academic Skills Resource and UoH Academic Integrity Resource module in Brightspace. If you experience difficulties with this assessment or with time management, please speak to the module tutor/s, your Personal Academic Tutor, or the School's Guidance Team. (sce.guidance@hud.ac.uk).
Requesting a Late Submission	You are reminded to 'back-up' your work as late submission requests will not be given for lost work, which includes work lost due to hardware and software failure/s. Late submission requests will only be approved if you can demonstrate genuine, unexpected circumstances along with independent supporting evidence (e.g. medical certificate) that may prevent you submitting an assessment on time. Submit your request for Late Submission via MyHud/MyStudies within 2 working days of the due date. Late submission requests, up to a maximum of 10 working days, but typically 1-5 working days, will be considered provided that there is appropriate evidence which clearly indicates reasons for the request. You will have 5 working days after submitting a request to provide the evidence. Failure to submit evidence will result in the request being rejected and your work being marked as a late submission (see below). If you are unable to submit work within the maximum late submission period of 10 days, contact the School's Guidance Team. (sce.guidance@hud.ac.uk), as you may need to submit a claim for Extenuating Circumstances (ECs).
Extenuating	An EC claim is appropriate in exceptional circumstances, when an extension is

Additional guidance information	
Circumstances (ECs)	not sufficient due to the nature of the request, or it concerns an examination or In-Class Test (ICT). You can access the EC claim form on the Registry website; where you can also find out more about the process. You will need to submit independent, verifiable evidence for your claim to be considered. Once your EC claim has been reviewed you will get an EC outcome email from Registry. If you are unsure what it means or what you need to do next, please speak to the Student Support Office– Room SJ1/01 An approved EC will extend the submission date to the next assessment period (e.g July resit period).
Late Submission (No ECs approved)	Late submission, up to 5 working days, of the assessment submission deadline, will result in your grade being capped to a maximum of a pass mark. Submission after this period, without an approved extension, will result in a 0% grade for this assessment component.
Tutor Referral available	YES
Resources	• Please note: you can access free Office365 software and you have 1 Tb of free storage space available on Microsoft's OneDrive – Guidance on downloading Office 365.

Coding Project

1. Assignment Aims

To develop an embedded system using C code running on a PIC16F882 microcontroller on your self-built tutorial board. You will apply your knowledge of the microcontroller architecture and programming skills developed during the module to produce a digital thermometer.

This assignment will test your abilities to apply the coding skills you have developed through the module in order to develop a novel solution a specific embedded systems application.

2. Learning Outcomes:

Knowledge and Understanding

a) Interpret manufacturers' data sheets to explain and compare the internal architectures of microcontrollers and their associated interfacing design strategies.

Abilities

c) Develop software solutions to microcomputer- and microcontroller-based system design problems through the use of appropriate design methods and development tools.

d) Develop hardware solutions to microcomputer- or microcontroller-based system design problems.

e) Evaluate the performance of microcomputer and microcontroller-based system designs and compare alternative solutions.

3. Assessment Brief

Digital Thermometer

Background

Digital thermometers are a useful tool with many applications both in industry and in the home. A common approach to temperature measurement is to use a thermistor which is a type of resistor whose resistance depends on strongly on temperature. Thermistors are typically constructed from metal oxides and come in two variants:

- Negative temperature coefficient (NTC) - resistance decreases with rising temperature
- Positive temperature coefficient (PTC) - resistance increases with rising temperature

Thermistors can generally operate over a temperature range of around -50 to 150 °C and are very stable over time which makes them idea for many common temperature measurement tasks. They can be used as part of a simple potential divider configuration to generate a temperature dependant voltage, as long as current flow is kept low enough minimise self-heating of the device. The main disadvantage of thermistors is that the resistance to temperature response is non-linear, so a calibration and correction must be applied.

Figure 1a A typical thermistor component

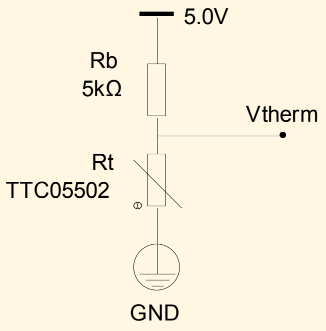


Figure 2b Simple potential divider setup to generate a temperature related voltage (Vtherm)

You will be using a TKS TTC05502 NTC thermistor in a potential divider configuration (see figure 1b) to generate a temperature dependant voltage, *Vtherm* which can be sampled by the PIC16F882 microcontroller via pin 25 (AN11). In this case *Vtherm* can be determined by the potential divider equation:

$$Vtherm = Vin\frac{Rt}{Rb + Rt}$$ 11* MERGEFORMAT ()

Solving for the thermistor resistance, Rt yields

$$Rt = \frac{Vtherm \cdot Rb}{Vin - Vtherm}$$ 22* MERGEFORMAT ()

The response of the TTC05502 is shown in figure 3 (note the y axis is on a log scale). The nominal resistance of the TTC05502 at 25 °C is 5kΩ.

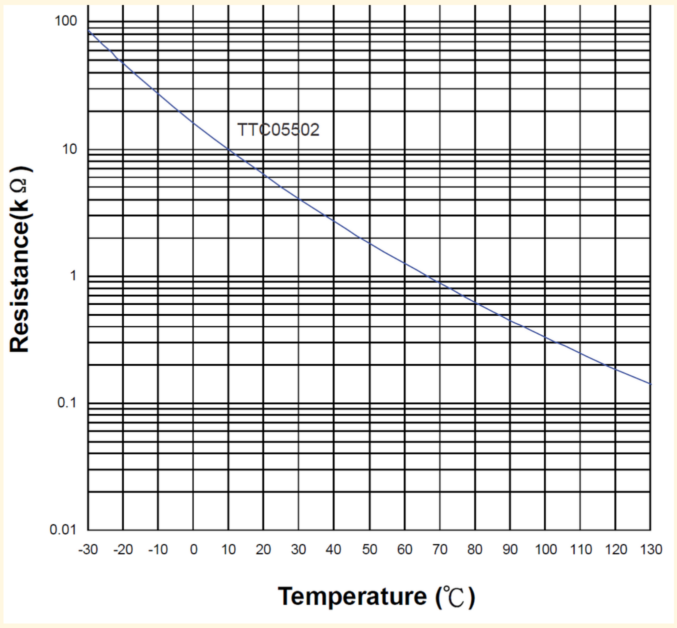


Figure 3 Resistance vs temperature response of the TKS TTC05502

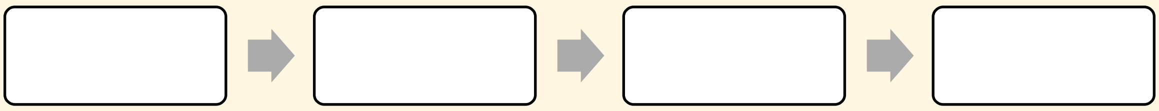
The non-linear response of NTC thermistors can be conveniently modelled to a good approximation using the Steinhart-Hart formula, where the temperature, T in Kelvin is related to the thermistor resistance, R as

$$T = \frac{1}{A + B \cdot \ln(R) + C \cdot [\ln(R)]^3}$$

33* MERGEFORMAT ()

For the TTC05502, the Steinhart-Hart coefficients are: A= 4.713 x 10^{-5}, B=4.760 x 10^{-4}, C= -1.151 x 10^{-6}

This model can be use as the basis for calibration of the TTC05502 and allows us to determine the current temperature using the process shown below:



Implementing logarithm calculation on a simple microcontroller such as the PIC16F882 is problematic because of the fixed-point architecture and limited memory. For this reason, it is common to implement complex mathematical operation using look-up tables.

For this project you will be provided with a library function (which must be used) to implement equation (3) using a look up table.

The digital thermometer must be able to display the current temperature in both °C and °F. The conversion can be made by using the following formula:

$$T_{\deg F} = (T_{\deg C} \times 1.8) + 32$$

44* MERGEFORMAT ()

Hardware Specification

The diagram below summarises the inputs and outputs to the PIC16F882 which will be used to implement the system.

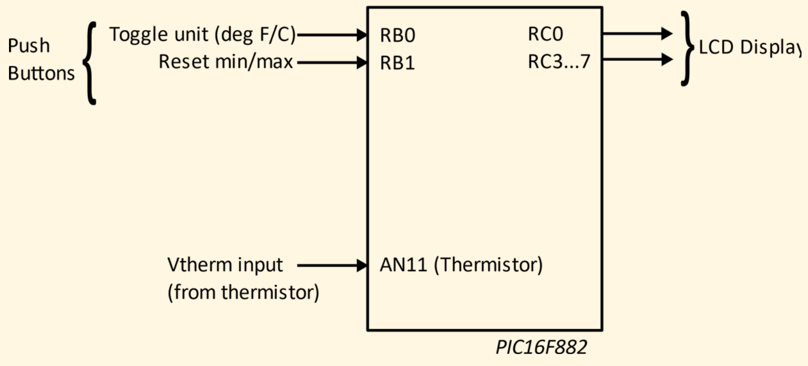


Figure 4 Schematic of the system inputs and outputs

Operational Specification

Overview

The embedded system for this project is based upon the PIC16F882 microcontroller and your self-built tutorial boards. You must connect a thermistor and bias resistor in the potential divider configuration (as shown as figure 2b) such that the temperature dependant voltage, *Vtherm* is applied to pin 25 which is configured as analogue input (AN11).

During operation the embedded system must do the following:

1) Continuously sample *Vtherm* at a rate of 1 Hz;
2) Convert the sampled *Vtherm* value into a temperature in °C (using the provided library function);
3) Display the current temperature on the top line of the LCD;
4) Calculate the maximum/minimum temperature recorded since start-up (or reset);
5) Display the maximum/minimum temperatures on the bottom line of the LCD.

Display format

The display should update at a rate of 1 Hz and be free of flicker or other display artefacts. Figure 5 shows required display format for the calendar, all temperatures are displayed in integer form.

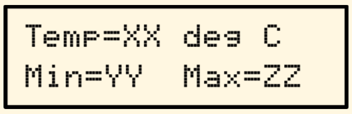

```
Temp=XX deg C
Min=YY  Max=ZZ
```

Figure 5 LCD display format for the digital thermometer where: XX is the current temperature; YY is the minimum recorded temperature; ZZ is the maximum recorded temperature.

Pushbutton inputs

The system must react to inputs from the pushbuttons on the tutorial board in the following manner:

- Switch 0 (RB0) – Change temperature unit from ° C to ° F
- Switch 1 (RB1) – Reset minimum/maximum recorded value;

Additional requirements

Interrupts

10 marks are assigned to the use of interrupts to implement certain functionality. Using interrupts will also result in a more responsive system better code. The following interrupts should be implemented:

- Timer 1 overflow interrupt (TMR1E/TMR1F) – For timing screen updates and temperature sampling.
- PORTB change Interrupt (RBIE/RBIF) – To detect pushbutton activity

Innovation

10 marks are assigned for the addition of innovative features that provide *useful* functionality that goes beyond the system specified above. Such innovation is left to your creative and programming skills, but the following list outlines some possible ideas (in order of anticipated difficulty):

- Calculate and display the mean temperature since startup;
- Add a decimal place of precision to the temperature readout;
- Use the tutorial board LEDs to produce a bar graph output where each LED represents a 10 degree temperature interval between 0 and 80 ° C.

1. <u>Marking Scheme</u>

Developed project code (100 marks total)

• Code functionality	(20 marks)
• Design/structure of the code	(20 marks)
• Ability to answer questions on code design	(20 marks)
• Code efficiency	(10 marks)
• Code layout and annotation	(10 marks)
• Use for interrupts	(10 marks)
• Innovative features	(10 marks)

Grading Rubric

Criterion / Mark	Excellent (16-20)	Good (12-16)	Fair (8-12)	Poor (0-7)	Marks
Code functionality	Full functionality demonstrated.	Majority of functionality implemented	Partial functionality implemented	Little or no functionality implemented	**20**
Design & structure of the code	Multi-parameter functions and return values to maximise code reuse. Careful consideration of variable scope to maximise data safety is evident throughout. Advanced use of loop variables and operators to minimise code repetition. Branching code implementation is logical and designed with clarity in mind. Effective and logical use of one or more library files demonstrated.	Several single-parameter functions are used effectively to enable substantial code reuse. Consideration given variable scope to maximise data safety but not consistent throughout. Effective use of loops to minimise code repetition. Branching code implementation is logical throughout. The ability to implement a library file demonstrated.	Some basic function usage with no parameters implemented. Little consideration given to variable scope to maximise data safety. Some limited use of simple loops to minimise code repetition. Branching code implementation may be confusing at times. Library file is either not present or not implemented correctly.	Function implementation is either very limited or non-existent. No consideration has been given to sensible variable scope to maximise data safety. Loop use is rudimentary and substantial code repetition is in evidence. No attempt to use a library file has been made.	**20**
Abilities to answer questions on code design	All lines of questioning are confidently addressed and and a total understanding about all aspects of the code design is demonstrated.	Most lines of questioning are address confidently but some aspects of the code design are not clearly articulated.	Lines of questioning are answered in a variable manner but some understanding of the code design is demonstrated.	Few or non of the lines of questioning are ansered effectively. Little or no understanding of the code design is demonstrated.	**20**
	Excellent (9-10)	**Good (6-8)**	**Fair (4-5)**	**Poor (0-3)**	
Code efficiency	Code is written very efficiently with very little room for further improvement. Substantial efforts have been made to reduce the code size without impacting on readability. Virtually no avoidable code repetition present. Variable type choice is at all times optimal.	Code generally efficient. Efforts have been made to reduce size of the code though some avoidable repetition is still present. Variable type choice is fairly optimal with some minor exceptions.	Code is either moderately efficient OR generally efficient with some major exceptions. Only minor efforts has been made to reduce size of the code. Unnecessary code repetition is noticeable throughout. Variable type choice is often non-optimal and wasteful.	Code is inefficient overall. There is a large amount of unnecessary code repetition brought about by a lack of consideration in the design. Variable type choice consistently non-optimal. Little or no consideration of good practice is evidenced with respect to developing efficient code.	**10**
Code layout & annotation	*All of the following attributes are implemented to a high standard:* Full header containing filename, author, version number and description. Consistent and neat indentation. Comprehensive yet concise commenting. Function descriptors.	*Most of the following attributes are implemented to a good standard:* Full header containing filename, author, version number and description. Consistent and neat indentation. Comprehensive yet concise commenting. Function descriptors.	*Some of the following attributes are implemented to a fair standard:* Full header containing filename, author, version number and description. Consistent and neat indentation. Comprehensive yet concise commenting. Function descriptors.	*Few/none of the following attributes implemented, to a poor standard:* Full header containing filename, author, version number and description. Consistent and neat indentation. Comprehensive yet concise commenting. Function descriptors.	**10**
Use of interrupts	Extensive interrupt usage demonstrated above that specified in the brief.	All suggested interrupts in the brief implemented correctly.	Partial implementation of interrupts in the brief.	No interrupts implemented.	**10**
Innovative features	Several highly innovative features implemented.	At least one highly innovative feature implemented.	Some minor innovative features implemented.	Either a very minor innovation feature demonstrated or none at all.	**10**