CSE321 Project 1

Assigned: September 18, 2020
Project Due: October 4, 2020, 11:59 pm EST
Note: October 4 will be an independent work day instead of a lecture to wrap this all up.

Read this entire document before getting started.

Objective:

You will be setting yourself up for success by creating a base reference guide and setting up your environment. By creating these components, you will be prepared for design implementation in future projects.

Part 1: Getting Started

Create a Word document file to contain all of your written work for this project. Name the file **CSE321_project1_*username***, where ***username*** refers to your UBIT username. That is, the part of your email address that comes before "@buffalo.edu". Put your name in the header and right-align it.

Your method of working on this file is up to you. We have a MS365 subscription that can be used as cloud storage/workspace via the browser, but you can also work locally on your copy of Word. Make sure you back up work regularly.

Part 2: Setting Up PlatformIO

Now you'll set up your IDE. You'll need to install it on your computer and provide evidence that it's installed. To get the IDE, go to Atom's website and get the appropriate version of the Atom IDE. Then install PlatformIO.

Part 3: Creating Your Code Template

Standardized templates improve the ease of collaboration and hand-offs. You will create a template that presents key information and structure to allow for consistent design approaches.

Best practices in programming establish that **all files have a commented segment at the top** that provides information about the file. This should contain the following information for the reader:

- Who wrote the code
- The file's purpose
- Any modules/subroutines
- If it corresponds to any assignments
- The inputs
    - *Note: for the main file this will be for the peripheral connections. Function-specific inputs should be noted in the comments with the function.*
- The outputs
    - *Note: for the main file this will be for the peripheral connections. Function-specific outputs should be noted in the comments with the function.*
- Any constraints
- Any sources or references used (**with links**)

Create a new project in Atom/PlatformIO with the MBED Framework, and edit the *.cpp file that's generated to include a header comment. This is the starter code for the files in all of the projects. Use the examples seen in class and in the Getting Started video.

Expectations for your template are discussed in lecture 3, which covers formatting as well. As seen in the University of Utah site referenced that day, each of the required pieces of information are clearly labeled in the header section. For the information that does not change, you will put that in now. For information/content that will change, you will leave it blank.
(Do not put an explanation of the information that will eventually be placed there like shown in the examples on the University of Utah website)

Add a screenshot of the program to the **CSE321_project1_*username*** Word document. Make sure that the screenshot shows the code template file and that Atom's file tree is open (ctrl+/ or cmd+/).

Save a copy of the code template as **CSE321_project1_*username*_template.cpp** by right-clicking on the "main.cpp" file and then click "Rename". You will submit this file on UB Learns along with the rest of your submission.

Grading:

	Unacceptable	**Acceptable**	**Exemplary**
Template Content	Significant issues or not done or not correct file format [0 Points]	Slightly some fields missing or inappropriate [9 Points]	Appropriate and complete [15 Points]
Screenshot of template	Significant issues or not done or not correct file format [0 Points]		Appropriate and complete [10 Points]
Screenshot of project tree	Significant issues or not done or not correct file format [0 Points]		Appropriate and complete [10 Points]

Part 4: Setting Up Git

Version control is essential when working or collaborating on projects, and it helps when projects are completed over a period of time, too. Some jobs or careers might even require an applicant to demonstrate their understanding of version control, so it's a useful skill to have.

For this part, you'll need to create a GitHub account. Go to GitHub's website and follow the instructions to set up an account.

After your account is created, you'll make a private repository using the GitHub classroom. This repository will be used for the course to showcase your design and implementation skills. it will be a good way to archive your work for the future as well. Importantly, **it needs to remain private through the end of this course**.

Once it's created, you'll need to fill out some information. In the "About" section, put a summary of what this repository will contain. Then make 3 folders named "Project 1", "Project 2", and "Project 3".

Place a copy of this instruction sheet in the "Project 1" folder along with a copy of your code template and the other specified documents.

In the **CSE321_project1_*username*** Word document, write down the username for your GitHub account that you used to create your repository. Without this we can't find your repository and you won't receive credit for your work.

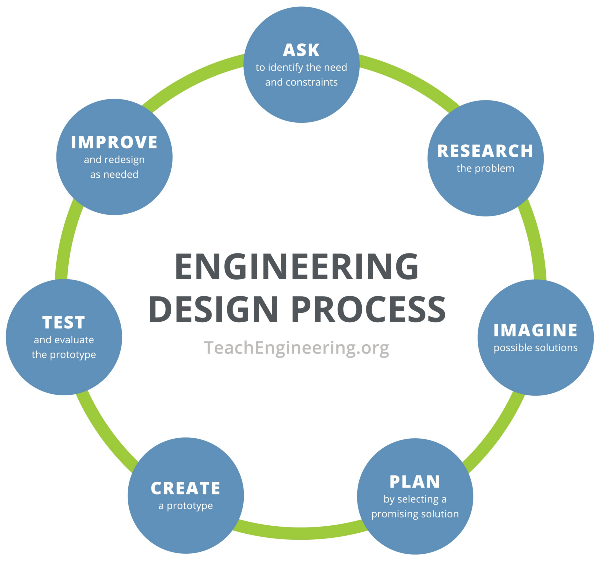
Throughout the term, it is up to you how you want to connect to your repository. At a minimum, there is always the ability to push/upload your files via the website.

Grading:

	Unacceptable	**Acceptable**	**Exemplary**
Username provided (graded on the UB Learns submission)	Significant issues or not done or not correct file format [0 Points]		Appropriate and complete, able to access repository [15 Points]
"About" section	Significant issues or not done or not correct file format [0 Points]	Slight details missing or inappropriate [4 Points]	Appropriate and complete [10 Points]
Folders	Significant issues or not done or not correct file format [0 Points]	Minor issues with folder naming or missing a folder [4 Points]	All accounted for and with correct names [10 Points]
Content in "Project 1" Folder	Significant issues or not done or not correct file format [0 Points]	Some elements missing or with incorrect naming [9 Points]	Appropriate and complete [15 Points]

Part 5: Establishing Good Planning Practices

Preparing for projects and presenting information and solutions are important skills to have. In this project, you'll get experience in the initial stages of planning a project, which will prepare you for future projects in the course and in your career.


ASK
to identify the need and constraints
RESEARCH
the problem
IMAGINE
possible solutions
PLAN
by selecting a promising solution
CREATE
a prototype
TEST
and evaluate the prototype
IMPROVE
and redesign as needed
ENGINEERING DESIGN PROCESS
TeachEngineering.org

Specifically, in this project you'll be focusing on the "Ask", "Research/Imagine" and "Plan" steps of the engineering design process for the problem below.

Problem:

> An IoT device is needed for controlling traffic on campus based on geese proximity. These are special geese, and they need to stay safe. The device will be programmed with a standard embedded OS and will make use of sensors for detecting traffic and geese. The traffic is controlled by a single light that will stop traffic in all directions, when needed, to protect the geese by turning red. When traffic can flow, the light blinks red and is treated as a stop sign.

The way you choose to present your answers to the first few stages of the design process is up to you, but it must be professional. In this context, that also means being concise and focused (it does not necessarily need to be in paragraphs, but you should not have any "fluff").

Your plan can be presented as a task list or in a flow chart-like structure (it's your choice). **It must be constructed digitally (no hand drawings or writings).**

Add this work to your **CSE321_project1_*username*** file that also has the solution elements from the previous parts of this project.

Grading:

	Unacceptable	**Acceptable**	**Exemplary**
Ask	Significant issues or not done or not correct file format [0 Points]	Slightly some fields missing or inappropriate [9 Points]	Appropriate and complete [15 Points]
Research/Imagine	Significant issues or not done or not correct file format [0 Points]	Slightly some fields missing or inappropriate [9 Points]	Appropriate and complete [15 Points]
Plan	Significant issues or not done or not correct file format [0 Points]	Slightly some fields missing or inappropriate [9 Points]	Appropriate and complete [15 Points]
Professionalism	Significant errors or not correct file format [0 Points]	Minor errors. [5 Points]	Professional, grammar, and spelling are appropriate. Everything is digitally created. [10 Points]

Part 6: Establishing Good Coding and Documentation Practices

A good exercise for understanding the importance of good communication in programming is fixing a disastrous bit of code with a poorly written README file.

Open the provided code and clean it up. This means you may need to address the following issues.

- Professional Free Field layout
- Create an appropriate header
- Add comments
    - Any line that is not a very clear self-commenting operation must have a comment
    - Fix comments
- Make variable names appropriate for a professional setting
    - Choose self-commenting names if possible
- Remove unnecessary and inappropriate elements

Some additional details on comments

- Some comments are correct
- Some comments need updating/fill in the blank
- Some are badly worded
- Some are totally inappropriate
- Some tell you what to do to the code to make the appropriate correction
- Some do not change the content of! The line numbers are based on the initial provided code. You can rearrange them as it makes sense.
    - 14
    - 50
    - 62
    - 89
    - 91
    - 97

The program must still run when you are done with the same behavior it had when you started.

Save this file as **CSE321_project1_*username_corrected_code.cpp***

README files are critical to collaboration, deployment, and continuous improvement. There are many variations on the content, but it needs to be appropriate for the project. For this assignment you will be doing a plain text README rather than the markdown style which is often seen (you will do markdown in the future). You can reference some tips on README contents and formatting from the University of Oklahoma and this guide from Akash Nimare.

Update the README file. This means you may need to address the following issues.

- Professional Free Field layout

- Overall grammar and language
- Relevant and correct content
- Clear and correct specifications
- Correct constraints, parameters, and names

This program makes use of some elements (specifically API functions) that have not been covered yet. These are tools with specific purposes. You will need the information here to create appropriate comments in the code and the **readme.md**

- `Thread`
  - Allows things to be configured with scheduling and priority management. It is a block of code that will execute based on an event.
  - There are methods to access information with this controller and to control it. Specifically in this project you will see: `get_state` and `start`
- `DigitalOut`
  - Allows you to set up a GPIO pin as an output, doing all that initialization stuff.
- `InterruptIn`
  - This creates a reference to an interrupt with a variable name to allow additional interaction. This interrupt is triggered by the button, with actions on both rise and fall of the signal.
  - There are methods to configure the interrupt, specifically: `rise` and `fall`.

Do not change these terms/lines in the code. You can change other elements that are adjacent to them if appropriate. You can rearrange and move them still as appropriate.

```
#include "mbed.h"
Thread
DigitalOut
InterruptIn
.start
.rise
.fall
.get_state
thread_sleep_for(500)
thread_sleep_for(2000)
printf
```

Save this updated readme as **CSE321_project1_*username_readme.*[*md*]()**

Push these files to the “Project 1” folder in your repository and submit them on UB Learns with your final submission.

Grading for Code:

	Unacceptable	**Acceptable**	**Exemplary**
Code Free Field Setup Corrected	Significant issues or not done or not correct file format [0 Points]	Slightly some fields missing or inappropriate [9 Points]	Appropriate and complete [15 Points]
Variable names corrected	Significant issues or not done or not correct file format [0 Points]	Slightly some fields missing or inappropriate [9 Points]	Appropriate and complete [15 Points]
Correction of initial comments	Significant issues or not done or not correct file format [0 Points]	Slightly some fields missing or inappropriate [9 Points]	Appropriate and complete [15 Points]
Additional comments added for clarification	Significant issues or not done or not correct file format [0 Points]	Slightly some fields missing or inappropriate [9 Points]	Appropriate and complete [15 Points]
Builds	No [0 Points]		Yes [10 Points]

Grading for README

	Unacceptable	**Acceptable**	**Exemplary**
Completion of file	Significant issues or not done or not correct file format [0 Points]	Slightly some fields missing or inappropriate [9 Points]	Appropriate and complete [15 Points]
Removal and updating of provided content	Significant issues or not done or not correct file format	Slightly some fields missing or inappropriate	Appropriate and complete [15 Points]

	[0 Points]	[9 Points]	
General language/writing	Significant issues or not done or not correct file format [0 Points]	Slightly some fields missing or inappropriate [9 Points]	Appropriate and complete [15 Points]
About section	Significant issues or not done or not correct file format [0 Points]	Slightly some elements missing or inappropriate [2 Points]	Appropriate and complete [5 Points]
Features section	Significant issues or not done or not correct file format [0 Points]	Slightly some elements missing or inappropriate [2 Points]	Appropriate and complete [5 Points]
Required materials section	Significant issues or not done or not correct file format [0 Points]	Slightly some elements missing or inappropriate [2 Points]	Appropriate and complete [5 Points]
Resources and References section	Significant issues or not done or not correct file format [0 Points]	Slightly some elements missing or inappropriate [2 Points]	Appropriate and complete [5 Points]
Getting Started section	Significant issues or not done or not correct file format [0 Points]	Slightly some elements missing or inappropriate [2 Points]	Appropriate and complete [5 Points]
File section overall	Significant issues or not done or not correct file format [0 Points]	Slightly some elements missing or inappropriate [2 Points]	Appropriate and complete [5 Points]
Declarations	Significant issues or not done or not correct file format [0 Points]	Slightly some elements missing or inappropriate [2 Points]	Appropriate and complete [5 Points]
APIs and Elements	Significant issues or not done or not correct file format [0 Points]	Slightly some elements missing or inappropriate [2 Points]	Appropriate and complete [5 Points]

Custom functions	Significant issues or not done or not correct file format [0 Points]	Slightly some elements missing or inappropriate [2 Points]	Appropriate and complete [5 Points]

Part 7: Submission

One of the most important aspects of a project is the submission of work. There are several things to consider before you submit. The points for this section will be awarded based on the submission and how well you follow the submission requirements.

UB Learns files to be uploaded:

- PDF version of your **CSE321_project1_*username*** word document
- **CSE321_project1_*username*_template.cpp**
- **CSE321_project1_*username_corrected_code.cpp***
- **CSE321_project1_*username_readme.md***

GitHub repository files to be pushed to the “Project 1” folder:

- Copy of project instructions
- PDF version of your **CSE321_project1_*username*** word document
- **CSE321_project1_*username*_template.cpp**
- **CSE321_project1_*username_corrected_code.cpp***
- **CSE321_project1_*username_readme.md***

Submissions in both locations count towards your grade. GitHub related grading will have a separate spot in the gradebook.