Feasibility Model Design

F2019 – Edit this document into a deliverable.

|  |  |  |  |
| --- | --- | --- | --- |
| Lab Section: | 5 | Group: | 16 |

# System-Level Design



Figure 1: Team 16 System level design

## Project Design Requirements

In PD 21 you learned about engineering requirements. they fall into three major categories, as follows:

1. **Functional requirements** are quantities that specify the performance of a design. They are related to the functions of the design, identified as answers to the question, "What does it do?" For example, a functional requirement for a coffee maker may specify the time required to brew a pot of coffee, a DC power supply may specify its maximum voltage, and a vehicle alarm system may specify how much noise it makes when it is set off
2. **Non-functional requirements** specify characteristics of the design that are not performance based. Theses are typically features or qualities that are desirable to the client. For example, ease of use, ease of manufacturing, and use of recycled materials.
3. **Constraint requirements** place limits on the design space, and often reflect budget or other project limitations. For example, cost, weight, and noise.

The basic form of most of these requirements is the same: a short description, followed by a relationship (equals, less than, or greater than) and a value.

**State three to five major Functional Requirements that your project must meet to successfully solve your problem statement.**

Functional:

* Must have at least two functional axis of movement.
* Must initialize to center of axis range, for both axis. (start x = width / 2, start y = height / 2)
* Must support coordinates inputs that are positive and/or negative.
* Must actuate LED indicator upon reaching range limits.
* Must stop appropriate motor upon reaching range limits.
* Must allow user to input coordinate values via push button, keypad or UART USB.
* Must allow user to start movement sequence via push button.
* Must display bar-graph reflecting current movement progress on LCD display.

Non-functional:

* Must easy to use (input coordinates, start sequence, etc.).
* Must reach coordinates within reasonable time delay.
* Must travel in one smooth motion (no jitter, grinding, skipping).
* Must reach targets with reasonable accuracy and/or precision
* Must be energy-efficient.

Constraint:

* Must cost less than 100$ per unit.

## Project Sensors and User Inputs

* List the types of sensors and user inputs you may require (light, sound, temperature, magnetic field).
* For each sensor and user input, list how you will connect it to the MCU, including additional interface components, if needed.
* 4x limit switches, implemented as push buttons, connected to MCU using digital input pins
* Keypad for entering coordinate values, connected to MCU using digital inputs pins

## Project Actuators and Indicators

* List the types of actuators and indicators you may require (e.g. light, sound, mechanical motion)
* For each actuator and indicator, list how you will connect it to the MCU, including additional interface components, if needed.
* 4x LEDs as limit indicators, connected to MCU using digital outputs pins
* 2x Stepper motors for x-y movement, connected to stepper motor driver to interface with MCU
* LCD display, connected to digital outputs (on board)

## Project MCU Peripherals

* List the resources inside the MCU that could be used to implement your project (e.g. ADC, timers, interrupts, GPIO functions).
* List parameters that the software running on the MCU might require.
* 2x timers, for controlling stepper motor pulses
* 2x interrupts, for stopping motor control upon reaching limit switches
* UART USB, for alternate coordinate entering method, connected to MCU through serial interface

## Project Testing Methodology

* For each sensor, user input, actuator, indicator, and MCU peripheral listed above, state how you will verify that each one is functioning as expected (a table may be helpful)
* State how you will validate that each Project Design Requirement has been met

In order to test all of the following components, we will create a diagnostic utility program. In the table below, we outline the functionalities that this program will include, in order to verify each component individually. This utility will be used to verify proper functioning of **hardware** components. In the table blow, this utility will be referred to as “diagnostic”.

Additionally, we will have an integration test suite that verifies that the proper registers are being written to when certain inputs are given. This suite will only verify **software** logic. This test suite will be referred to as “integration”.

|  |  |
| --- | --- |
| Component | Verification method |
| Limit switches (push buttons) | Diagnostic: listen for limit switch button inputs, print out limit switch id on button press.  Integration: when button register has values corresponding to button inputs, ensure interrupt is triggered, LEDs should be turned on. |
| Keypad | Diagnostic: listen for keypad button inputs, print out key that was pressed id on button press.  Integration: ensure that LCD is being updated with correct input text. |
| Limit indicators (LEDs) | Diagnostic: turn on specific LED and ensure it lights up.  Integration: n/a. |
| Stepper motor | Diagnostic: make motor step *n* number of times and ensure motor turns appropriately.  Integration: Ensure proper calculations of number of steps necessary to reach destination. |
| LCD display | Diagnostic: send arbitrary string to LCD, ensure string is displayed properly.  Integration: Test states and ensure appropriate information is displayed. (Ex. while in input mode, displays coordinates, while moving display progress…) |
| Timers | Diagnostic: input time value, ensure timer waits that amount of time before outputting message.  Integration: See stepper motor tests. |
| Interrupts (limit) | Diagnostic: trigger interrupt and have interrupt handler print to console.  Integration: See push button tests. |
| UART USB | Successfully sends messages over USB. Will be tested through all other components tests. |

# Feasibility Model Diagram and Software Flowchart (High-Level)

A simplified example is shown in Figure 2 and Figure 3. **Replace these figures with high-level block diagrams of your system.**



Figure 2: Feasibility Model Design



Figure 3: Software Flowchart

## Initial Bill of Materials

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| LAB SESSION NUMBER: 5 | | TEAM NUMBER: 16 | | | Order #: 1 |
| **PART NAME** | **Distributor Part No** | **ECE 398 DipTrace Part** | **RIGIDWARE SKU** | **RIGIDWARE PRICE (without TAX)** | **QUANTITY** |
| MODULE - Stepper Motor Driver | Amazon Distributed Longmire ULN2003 Stepper Driver Board |  | 4916796 | $2.30 | ~~2~~ 1 (Due to limited quantity) |
| MODULE - Keypad 4x4 | Sparkfun COM-14662 |  | 4915977 | $6.00 | 1 |
| MODULE - Stepper Motor 5V | Amazon Distributed Stepper Motor |  | 4916741 | $2.30 | ~~2~~ 1 (Due to limited quantity) |
| COMPONENT - PB Switch SMT | Digikey CKN9112DKR-ND | UPARTPBSW | 4924001 | $0.55 | 6 |
| COMPONENT - RED LED - Diffused 5mm) | Digikey 1497-1031-ND | QPARTR5D34 | 4916296 | $0.40 | 4 |
|  |  |  |  | **20.10** |  |