# Project Design Requirements

1. **Functional Requirements**
   1. Need a virtual comm terminal to set the system parameters in SETUP MODE and then starting the system in RUN MODE.
   2. The Window/Door OPEN/CLOSE sensors senses if its state is OPEN or CLOSED.
   3. Doors must also monitor approaching persons (independently) and then using distance sensors to activate external lighting.
   4. Security System is armed or disarmed through a user keypad using a four-digit numeric combination or through comm terminal. (Password set through Comm Terminal)
   5. User has time determined by the **Activation Delay** (set by Comm) to disarm system once detected on premises.
   6. An **alarm situation** is triggered when any sensor is in OPEN state after system is armed. System must then activate an external Audible Alarm after Activation Delay.
   7. Security System must have Status Display implemented using an LCD screen showing time in 24-hour format. Must indicate mode (SETUP or RUN MODE). In SETUP mode, display all parameter names and values. In RUN MODE display all 6 OPEN/CLOSE sensor states (OPEN or CLOSED).
   8. LEDs must also indicate all sensors statuses (ARMED, DISARMED, ALARM) to more easily catch user attention (separate from LCD. Include distance sensors). In an alarm situation, ALARM LED must start flashing red.
   9. The Home Security System must be able to monitor the various sensors on its own if the COMM TERMINAL is not **used or powered**. COMM terminal is only for changing the system’s settings.
   10. Comm terminal must manage the following settings in SETUP MODE:
       1. Put Home Security System in SETUP Mode or RUN Mode.
       2. Set Time of day (hh:mm:ss).
       3. Keypad combination (four digits).
       4. Window Sensor 1 Arm/Disarm.
       5. Window Sensor 2 Arm/Disarm.
       6. Window Sensor 3 Arm/Disarm.
       7. Window Sensor 4 Arm/Disarm.
       8. Door Sensor 1 Arm/Disarm.
       9. Door Sensor 2 Arm/Disarm.
       10. Front Distance Sensing distance (1,2,3, or 4 meters).
       11. Front Distance Sensor Arm/Disarm.
       12. Back Distance Sensing distance (1,2,3, or 4 meters).
       13. Back Distance Sensor Arm/Disarm.
       14. Alarm Activation Delay (30, 45, 60, or 120 seconds).
2. **Non-functional Requirements**
   1. Distance sensors shall work within 1-4 meters.
   2. Comm system will be UART based.
   3. System must monitor access of up to 4 ground-floor windows and 2 door entrances.
3. **Constraint Requirements**
   1. Only parts from the Proteus Libraries may be used in this project.
   2. The Sensors, Actuators, User Inputs, and User Outputs (Indicators) will be connected to the Prototype Adapter Board through wiring harnesses to connectors on that board.
   3. The PCB shape must be as per the PCB layout template in ECE 298 Lab 4.
   4. The Prototype Adapter Board PCB has receptacle connectors to mate with the development board. They must not be changed in type or physical location.
   5. The final schematic and PCB design must include suitable decoupling capacitors for power supply voltage filtering for the PCB Adapter board (Bulk Decoupling) and for each IC (Local Decoupling).
   6. This project must employ voltage level translations (3.3 V → 5 V, 5 V → 3.3 V) for signals between the STM32 MCU (3.3 V) and 5 V devices where appropriate (refer to the part datasheets). Must start with 12V from wall, however.
   7. Must use a relay to control door lights and audible alarm. Connection to the external device must be isolated from all other electronic circuitry.
   8. For the External Lighting, the relay external connections must support a DC current connection minimum of 10 amps.
   9. For the Audible ALARM, the relay’s external connection must support a DC current connection minimum of 3 amps.

# Project Considerations for I/O

## **Project Sensors and User Inputs**

**Door/Window Sensors:** Using a general switch from ECE298 library (ECE298\_GEN\_SWITCH). Switch open = door/window closed. Switch closed = door/window open. Completed circuit can then go to pin on MCU to signal door/window open. MCU can process signal to update LCD and/or trigger alarm state if armed. Will need to convert voltage to range of MCU.

**Door Distance Sensors:** HCSR04 sensor. Can use MCU to convert pulse width of sensor output to distance and then drive external lights if in range set by virtual comm terminal. Will need to convert voltage to range of MCU.

**Keypad:** Using KEYPAD-PHONE. We know what button is pressed if we know what voltage corresponds to each column and what row the voltage comes from. MCU interprets this and stores in a register and waits until all 4 digits are pressed. All voltages used will already be within MCU’s limits.

**Comm Terminal:** Will use proteus’ virtual comm terminal. Used to communicate RUNMODE params or other data to a UART compatible device like the MCU. This terminal outputs 5V so will need to convert to 3.3V for serial communication.

## Project Actuators and User Outputs

**Status LEDs:** Will be using RGBLED-CC. The MCU will output window/door status to LEDs (3.3V) using BLUE/GREEN/RED for ARMED/DISARMED/ALARM.

**Door Lights:** Will be using generic ECE298 led (ECE298\_GEN\_LED). MCU will turn on LED once distance calculated from door sensor is within distance set by the comm terminal. (Actual user could change LED to anything).

**LCD:** Will be using the MILFORD-4X20-BKP. Assuming MCU can also output in UART (serial) format, it will display the time, mode, and all sensor statuses using its serial interface.

**Relay:** Will be using generic proteus relay. Will detect high signal from MCU to close a switch to let some other signal flow.

**Alarm:** Will be using proteus buzzer. MCU will activate buzzer after door/window open in ARMED state and after the activation delay. (Actual user could change this to anything).

## Project MCU Internal Resources

**Resources**

* We are going to need to use the ADC for the keypad as it uses analog voltages 1V, 2V, 3V and 1.45V for idle.
* We will need to use an internal timer to count time of high signal for door distance sensors and for the clock of the LCD.
* GPIO pins will be used for reading digital/analog input and sending response through output pins.
* If a pin detects a certain state such as a keypress or sensor update, an interrupt will be used to act on that data.

**Parameters**

* OPEN/CLOSE params for all doors/windows sensors. Need this data to determine whether windows or doors are open or closed. If OPEN and in ARMED state, change to ALARM state and sound alarm after delay. If OPEN and in DISARM state, nothing.
* ARMED/DISARM/ALARM params for doors/windows and distance sensors. This is to determine the state of all these objects.
* Keypad combination param. Need this to confirm if entered combination is equal to the one set.
* Activation delay param. Need this for the delay to start buzzer after ALARM state triggered.
* Time of day param. Need this to update time on LCD.
* System Mode (SETUP/RUN) param. Params can only be set in SETUP mode while those settings are used in RUN mode.

# Device Testing Methodology

## Device 1 – ECE298\_GEN\_SWITCH

In proteus, the switch is controlled via an open/close button which simulates a window/door opening/closing. We can observe the device’s functions by detecting the current through the switch.

A picture containing diagram

Description automatically generated

In the above example, we can see the switch working as expected. Creates open circuit when switch is open as shown by the 0A.

A picture containing diagram

Description automatically generated

In the above example, we can see the switch working as expected. Creates closed circuit when switch is closed as shown by the 500A. This also means that the switch has a very low nominal resistance of 0.1 Ohms which is ideal for reducing voltage drop as much as possible in our system.

5V was used as stimulus since all devices in our system except the MCU will be using that voltage. This 5V will be converted to 3.3V to be processed by the MCU (keep OPEN/CLOSED params updated).

## Device 2 – HCSR04

In proteus, the distance the sensor detects (in cm) can be adjusted with the up/down arrows on the device. We can observe the device’s functions by using the MCU to measure the pulse width from the ECHO output to determine the distance.

Diagram, schematic

Description automatically generated

Diagram, schematic

Description automatically generated

In the above examples, a 10us 5V pulse to the TR port is used to poll the distance from the sensor from the ECHO port. The output is a 5V signal where the distance is calculated from the pulse width. From here we can see the sensor is working correctly using the following formula: (high level time x (340m/s) / 2). Tested values from 1m to 4m as that will be the range of detection in our design. Example 1 expected output: 1m, actual output: 0.985m. Example 2 expected output: 4m, actual output: 3.95m. Considering this was measured by hand and not by the MCU, this level of accuracy will be fine. Again, 5V supply was used which is what this device requires to operate. (Oscilloscope would be MCU).

## Device 3 – KEYPAD-PHONE

A screenshot of a computer

Description automatically generated with low confidence

The device’s functions are controlled with a press of a button on the keypad. Each button will be able to be distinguished by the MCU by determining what voltage level each of the four pins (ABCD) are at. Here is the table to explain the I/O. Arbitrary 1V, 2V, and 3V inputs were using to distinguish rows. 1.45V is resting voltage (no buttons pressed). All voltages are below 3.3V to ensure compatibility with the MCU. (Oscilloscope would be MCU and will interpret these different voltages on different pins as button presses). Device works as expected and we can output 3 different voltage levels on 4 different pins for a total of 12 buttons. This then satisfies our need for a keypad.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Action | **Port A (Column 1)** | **Port B (Column 2)** | **Port C (Column 3)** | **Port D (Column 4)** |
| **Button 1 Pressed** | **1V (Row 1)** | 1.45V (N/A) | 1.45V (N/A) | 1.45V (N/A) |
| **Button 2 Pressed** | **2V (Row 2)** | 1.45V (N/A) | 1.45V (N/A) | 1.45V (N/A) |
| **Button 3 Pressed** | **3V (Row 3)** | 1.45V (N/A) | 1.45V (N/A) | 1.45V (N/A) |
| **Button 4 Pressed** | 1.45V (N/A) | **1V (Row 1)** | 1.45V (N/A) | 1.45V (N/A) |
| **Button 5 Pressed** | 1.45V (N/A) | **2V (Row 2)** | 1.45V (N/A) | 1.45V (N/A) |
| **Button 6 Pressed** | 1.45V (N/A) | **3V (Row 3)** | 1.45V (N/A) | 1.45V (N/A) |
| **Button 7 Pressed** | 1.45V (N/A) | 1.45V (N/A) | **1V (Row 1)** | 1.45V (N/A) |
| **Button 8 Pressed** | 1.45V (N/A) | 1.45V (N/A) | **2V (Row 2)** | 1.45V (N/A) |
| **Button 9 Pressed** | 1.45V (N/A) | 1.45V (N/A) | **3V (Row 3)** | 1.45V (N/A) |
| **Button \* Pressed** | 1.45V (N/A) | 1.45V (N/A) | 1.45V (N/A) | **1V (Row 1)** |
| **Button 0 Pressed** | 1.45V (N/A) | 1.45V (N/A) | 1.45V (N/A) | **2V (Row 2)** |
| **Button # Pressed** | 1.45V (N/A) | 1.45V (N/A) | 1.45V (N/A) | **3V (Row 3)** |

## Device 4 – Virtual Comm Terminal

Graphical user interface, application, Excel

Description automatically generated

We control the comm terminal via the command line window that pops up on run-time. Typing in it will communicate ascii through serial communication. As you can see here, another UART compatible device such as this LCD is able to interpret the incoming signals. This confirms the device is working as expected and should satisfy our requirements to interface with the MCU.

## Diagram Description automatically generated with medium confidenceDiagram Description automatically generated with medium confidenceDiagram Description automatically generatedDevice 5 – RGBLED-CC

This LED would be controlled by the 3.3V output pins on the MCU. Whatever signal is ‘high’ determines the output color. Which signal is high will be determined by whether the device that this LED is for is in ARMED, DISARMED, or ALARM state. Additional circuitry will be needed for the red light to flash but is not shown. The LED is driven by the 5V line. The generic ECE298 buttons are just used to simulate the MCU output on demand. This satisfies the need to signal what status of each door/window is.

## Device 6 – ECE298\_GEN\_LED

Diagram

Description automatically generated

This LED will be controlled by the 3.3V output pins on the MCU. If MCU calculates sensed distance is within params set by comm terminal, it outputs a ‘high’ signal to relay and is then powered by 5V line, shown by the blue color. This satisfies the need for door lighting.

## Device 7 – MILFORD-4X20-BKP

A screenshot of a computer

Description automatically generated with medium confidence

We will observe the device’s functions by hooking it up to a UART compatible device such as this virtual terminal or the MCU. This display should be large enough to show time, mode, and sensor status (all of which will come from the MCU). As shown by the image, the LCD can be easily programmed to show up to 80 characters. Therefore, this will satisfy our need to display all our relevant information.

## Device 8 – RELAY

Diagram

Description automatically generated Chart

Description automatically generated

Relay will be controlled via MCU output control signals. We can observe the device’s functions when a 3.3V is applied and closes the circuit to drive the device. This is behaving as expected since the device is completely off when MCU signal is off and device is on when MCU signal is on. (Button used to simulate MCU signal ON/OFF). This fulfills our requirements for isolating the lights and alarm from all other electronic circuitry. Additionally, these simulated relays have no limits to how much current they can handle, thus satisfying the 10A and 3A minimum for the external lights and alarm.

## Device 9 – Buzzer

Essentially the same mechanism as the light except with a 12V supply. We can observe its auditory functions by applying the MCU control signal for the alarm when a door/window is OPEN and in ARMED state after the activation delay. Because of this, it satisfies our requirements for an alarm.

# System-Level Design

