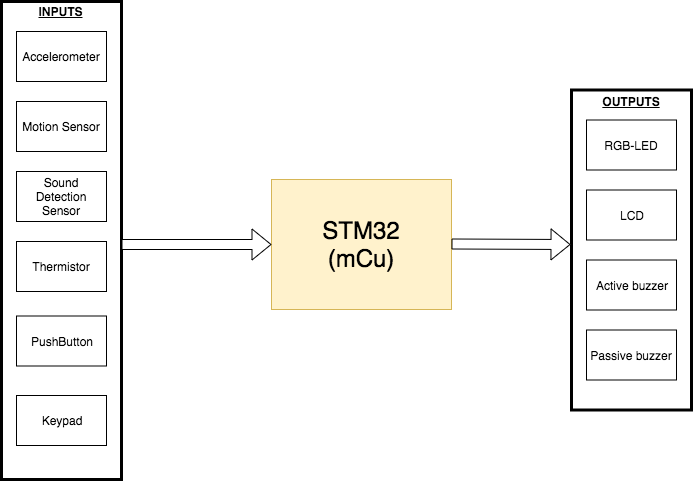
**Background**

Nordiska Museet has reached out to our team to solve an urgent problem. They have been presented the opportunity to host the largest diamond in the world. But unfortunately, their current alarm system is not up to par with the strict requirements set by the owner of the diamond. They must quickly upgrade their systems with a state-of-the-art alarm. None of the alarm systems available commercial-off-the-shelf (COTS) is good enough.

**Project Goal**

A high tech and advanced alarm system that will protect the diamond from everything.

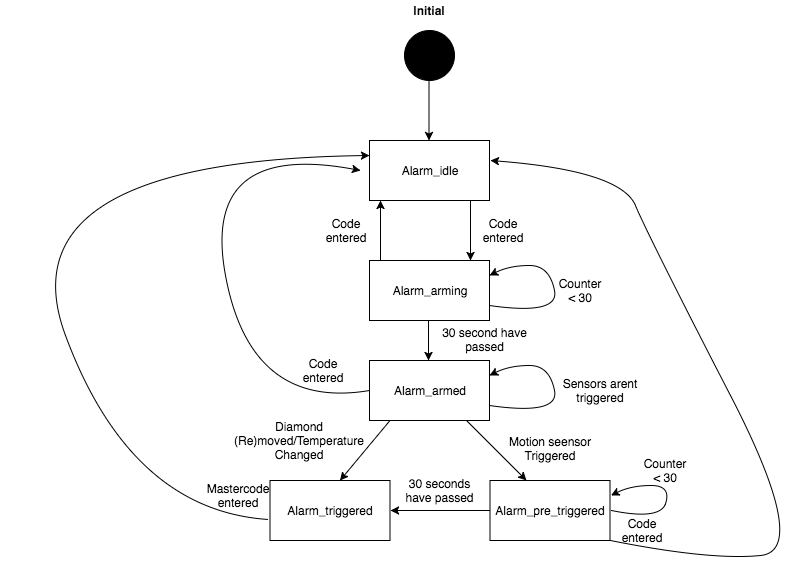
**OVERVIEW**

****

**COMPONENTS**

* STM32F401RE NUCLEO-64
* LCD 20X4
* Keypad 4X4
* PIR motion sensor
* RGB LED
* ADXL345
* Sound detection sensor
* Pushbutton
* 2x breadboards
* YWROBOT POWER MB
* 6.5-12v PSU/ BATTERY
* 2x Buzzers
* Wires
* Thermistor
* Resistors

**STATE MACHINE**



**FULFILLED REQUIREMENTS**

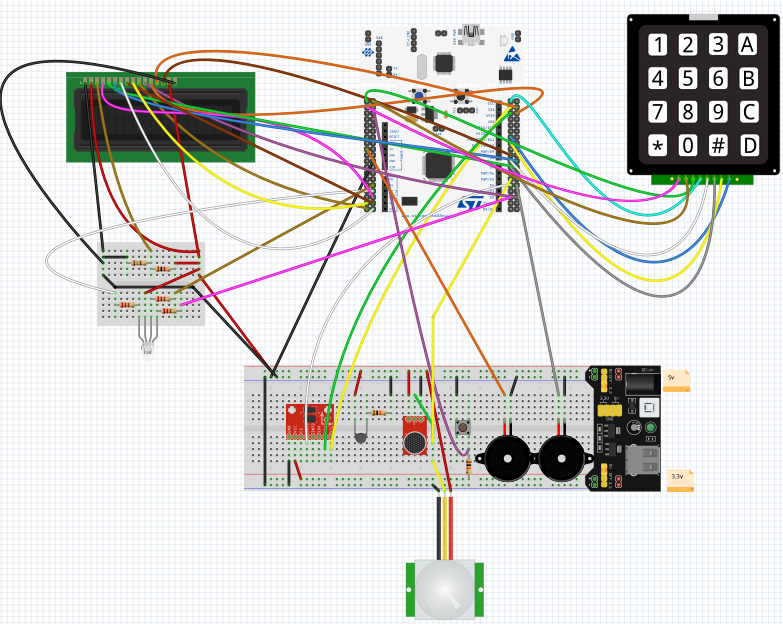
* An alarm system with an access panel consisting of a display and a keypad. A user interface is shown on the display presenting the status of the alarm and the possibility to deactivate/active the alarm with a 4 digits long code.
* LEDS indicating the current state.
* A PIR motion sensor to detect any thermal activities.
* A mechanical switch integrated beneath the diamond.
* Two buzzers
* Temperature sensor is added to measure any temperature changes and will then compare it to the set temperature.
* The set temperature is programmable via the keypad in the idle state by pressing the ‘A’ key on the keypad.
* An accelerometer is added and used in the activity interrupt mode. The accelerometer will generate an interrupt indicating that an intruder is detected.
* The threshold level for the accelerometer can be programmed by pressing the ‘B’ key on the keypad.

**ADDITIONAL FEATURES**

* Sound detection sensor
* The ‘D’ key works as a delete button when entering the code.
* The user code can be changed in idle state.
* A single RGB-LED instead of three LEDS.
* A bigger LCD-screen
* A cross-platform application displaying the actual state of the alarm and the temperature.

**PINOUT & SCHEMATICS**

|  |  |  |  |
| --- | --- | --- | --- |
| PIN LABEL | PIN NUMBER | PIN DESCRIPTION | GPIO MODE |
| LCD\_D0 | PC0 | LCD DATA PIN 0 | OUTPUT |
| LCD\_D1 | PC1 | LCD DATA PIN 1 | OUTPUT |
| LCD\_D2 | PC2 | LCD DATA PIN 2 | OUTPUT |
| LCD\_D3 | PC3 | LCD DATA PIN 3 | OUTPUT |
| LCD\_D4 | PC4 | LCD DATA PIN 4 | OUTPUT |
| LCD\_D5 | PC5 | LCD DATA PIN 5 | OUTPUT |
| LCD\_D6 | PC6 | LCD DATA PIN 6 | OUTPUT |
| LCD\_D7 | PC7 | LCD DATA PIN7 | OUTPUT |
| LCD\_RS | PB0 | LCD RS PIN | OUTPUT |
| LCD\_RW | PB1 | LCD RW PIN | OUTPUT |
| LCD\_E | PB2 | LCD ENABLE PIN | OUTPUT |
| PAD\_COLUMN\_1 | PA6 | KEYPAD COLUMN 1 | OUTPUT |
| PAD\_COLUMN\_2 | PA7 | KEYPAD COLUMN 2 | OUTPUT |
| PAD\_COLUMN\_3 | PA8 | KEYPAD COLUMN 3 | OUTPUT |
| PAD\_COLUMN\_4 | PA9 | KEYPAD COLUMN 4 | OUTPUT |
| PAD\_ROW\_1 | PC9 | KEYPAD ROW 1 | INPUT |
| PAD\_ROW\_2 | PC10 | KEYPAD ROW 2 | INPUT |
| PAD\_ROW\_3 | PC11 | KEYPAD ROW 3 | INPUT |
| PAD\_ROW\_4 | PC12 | KEYPAD ROW 4 | INPUT |
| ADC1\_IN0 | PA0 | THERMISTOR | INPUT |
| LED\_BLUE | PA10 | RGB-LED BLUE PIN | OUTPUT |
| LED\_RED | PA4 | RGB-LED RED PIN | OUTPUT |
| LED\_GREEN | PA1 | RGB-LED GREEN PIN | OUTPUT |
| I2C1\_SCL | PB8 | SCL TO ADXL345 | I2C |
| I2C\_SDA | PB9 | SDA TO ADXL345 | I2C |
| GPIO\_EXTI2 | PD2 | EXTERNAL INTERRUPT ADXL345 | EXTI |
| GPIO\_EXTI4 | PB4 | EXTERNAL INTERRUPT PIR/SOUND | EXTI |
| GPIO\_EXTI5 | PB5 | EXTERNAL INTERRUPT BUTTON | EXTI |
| TIM2\_CH1 | PA15 | BUZZER | PWM |
| ACTIVE\_BUZZER | PB13 | ACTIVE BUZZER | OUTPUT |



**CODE IMPLEMENTATION**

The code is divided into a couple of header/source files. The main part of the program is implemented in the alarm source file.

**Enums defined in the alarm header file**

|  |  |  |
| --- | --- | --- |
| Name | Values | Description |
| Alarm\_state | Alarm\_init, Alarm\_idle, Alarm\_arming, Alarm\_armed, Alarm\_PRE\_Trigged, Alarm\_Trigged, Alarm\_SetTemp, Alarm\_SetGyro, Alarm\_setCode | Contains all the states for the  state machine. |
| Key\_Code | Key\_No\_Pressed, Key\_Pressed,  Key\_OK,Key\_Wrong, Key\_A, Key\_B, Key\_C, Key\_D | This enum is used for the  function that returns the current state of the keypad. The function does also check if the right code is entered. |
| Sensor\_status | NONE, Motion\_Trigged,  Sound\_Trigged, Diamond\_Trigged | This enum is used to indicate  the states of the sensors. A variable of this type is declared and used together with the external interrupts to indicate the states of the sensors. |
| LED\_COLOR | L\_OFF, L\_RED, L\_YELLOW,  L\_GREEN | Contains all the different states the led can have in this program. |
| LCD\_Status | LCD\_NONE, LCD\_Unlocked,  LCD\_Locked, LCD\_Arming, LCD\_PRI\_Trigged, LCD\_Trigged, LCD\_SetTemp, LCD\_SetGyro, LCD\_SetCode | This enum is like the alarm enum and is used to control the printed text on the LCD. |

**Functions defined in alarm header file**

|  |  |  |  |
| --- | --- | --- | --- |
| Function type | Function name | Function parameter list | Function description |
| void | Alarm\_status | void | The alarm state machine is implemented in this function and every state calls a function. |
| Alarm\_state | A\_idle | TextLCDType lcd, uint8\_t setTemp,  uint8\_t code | This function is specifically for the idle state and relies on the function that checks the keypad status. |
| Alarm\_state | A\_arming | TextLCDType lcd, uint8\_t code | This function is specifically for the arming state and returns a different state either if the code is entered or 30 seconds have passed |
| Alarm\_state | A\_armed | TextLCDType lcd, uint8\_t setTemp,  uint8\_t code | This function is specifically for the armed state and returns a different state if a sensor is trigged or the code is entered |
| Alarm\_state | A\_Pre\_Trigged | TextLCDType lcd, uint8\_t code | This function is for the pre\_trigged state and contains a counter |
| Alarm\_state | A\_Trigged | TextLCDType lcd | If the user doesn’t enter code when the alarm is set of will this function run and handle the Trigged state |
| Alarm\_state | A\_setTemp | TextLCDType lcd, uint8\_t setTemp, | This function will run when the user enters the setTemp state and handles all the setups when the user want to change Temperature |
| Alarm\_state | A\_setGyro | TextLCDType lcd | Works like A\_setTemp but changes the threshold of the gyro instead |
| Alarm\_state | A\_setCode | TextLCDType lcd, uint8\_t code | Works like A\_setTemp and A\_setGyro but changes the code instead |
| Key\_code | Alarm\_code\_status | uint8\_t code | Handles the code input |
| void | update\_lcd | TextLCDType lcd, LCD\_status tmpS | Used to update the text print out on the LCD |
| void | lcd\_clearRow | TextLCDType lcd, uint8\_t row | Clears a row on the LCD |
| uint8\_t | check\_sensors | uint8\_t setTemp | Checks the status of the sensors and temperature |
| int16\_t | Read\_analog\_Temp | void | Reads the Analog value of the thermistor and converts it to a temperature in Celsius |
| void | Toggle\_Buzzer | void | Handles the passive buzzer |
| void | set\_Led | Led\_Color ld | Used to set the color of the RGB led |

**Functions defined in the keypad files**

|  |  |  |  |
| --- | --- | --- | --- |
| Function Type | Function name | Parameter list | Description |
| void | Keypad\_init | void | Set all the keypad pins to high |
| Int8\_t | Keypad\_read | void | Checks if a button on the keypad is pressed by setting one column at a time and then checks the input from the rows. Returns 99 if a button isn’t pressed |
| void | keypad\_setColumn | uint8\_t column | Set all the columns outputs to high and then resets a specific one. |
| uint8\_t | Keypad\_Checkrow | uint8\_t column | Checks the rows input value and returns the pressed keys value if a key is pressed. |

**Functions defined in ADXL345 files**

|  |  |  |  |
| --- | --- | --- | --- |
| Function type | Function name | Parameter list | Description |
| void | ADXL345\_writeRegister | uint8\_t regst, uint8\_t value | This function is used to write a value to a specific register |
| void | ADXL345\_ReadRegister | uint8\_t regst, uint8\_t \*data | This function will read from a specific register |
| void | ADXL345\_ReadRegisters | uint8\_t regst, int8\_t \*tmpData, uint8\_t dataLength | Almost same as “..readregister” but reads from multiple registers |
| void | ADXL345\_init | void | Initiates the gyro in measure mode. |
| void | ADXL345\_init\_interrupt | void | Initiates the gyro in interrupt mode |
| void | ADXL345\_clear | void | Used to clear an interrupt |
| void | ADXL345\_setThreshold | uint8\_t threshold | Used to set the threshold |