Report

Subject: Embedded Systems

Topic: User Interactions

**Student: Marusic Diana, FAF-171**

**Teacher: Bragarenco Andrei**

**Content**

1. **Domain**
2. **Component description**
3. **Implementation**
4. **Proof**
5. **Annex**

**Domain**

User interface design and user interaction design are two closely related disciplines. User interface focuses on the design of the visual interface, and user interaction focuses more on the design of the global interaction behavior of the system. Interaction design (IxD) is the branch of user experience design that illuminates the relationship between people and the machines they use. While interaction design has a firm foundation in the theory, practice, and methodology of traditional user interface design, its focus is on defining the complex dialogues that occur between people and interactive devices of many types-from computers to mobile communications devices to appliances.

Good interaction design effectively helps communicate a system’s interactivity andfunctionality, defines behaviors that communicate a system’s responses to user interactions, reveals both simple and complex workflows, informs users about system state changes, and prevents user error.

By hardware, user interaction devices can be divided in:

1. Input hardware:

* Buttons
* Keyboards/keypads
* Mice
* Game pads
* Joystick button for a pointing device

1. Output hardware:

* LEDs
* Monitors(screens)
* Speakers
* Graphic or character LCDs

1. Both input and output:

* graphical screen with touch sensing (touchscreen)

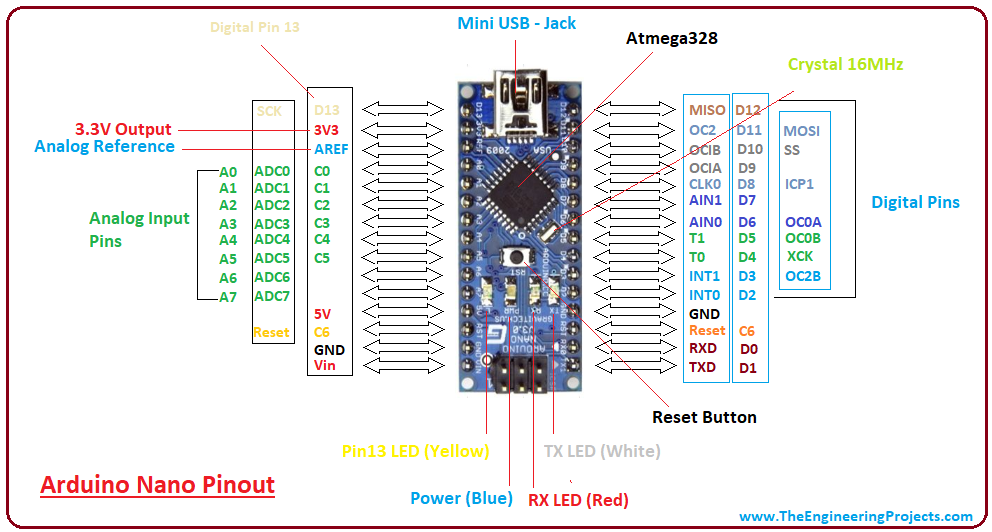
**Types of user interfaces:**

* command line (cli)
* graphical user interface (GUI)
* menu driven (mdi)
* form based (fbi)
* natural language (nli)

User interactions are an important part of almost every project, especially in embedded systems, because it usually requires the user to input data, set some parameters, turn on/off a device, setup, take decisions for systems to function and serve its purposes. At the same time, it is very important for the user to know the device state, receive feedback & errors and have a way of communicating and reading devices.

**Component description**

This section will describe each component and provide details about usage and advantages.

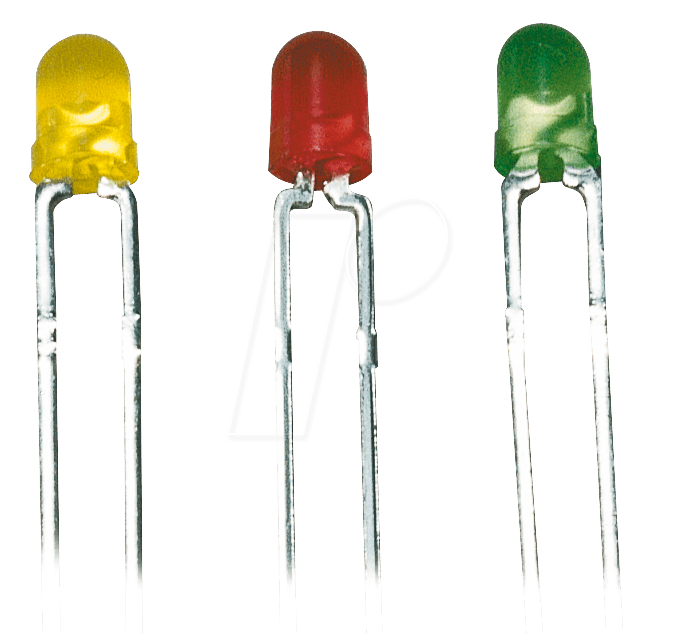
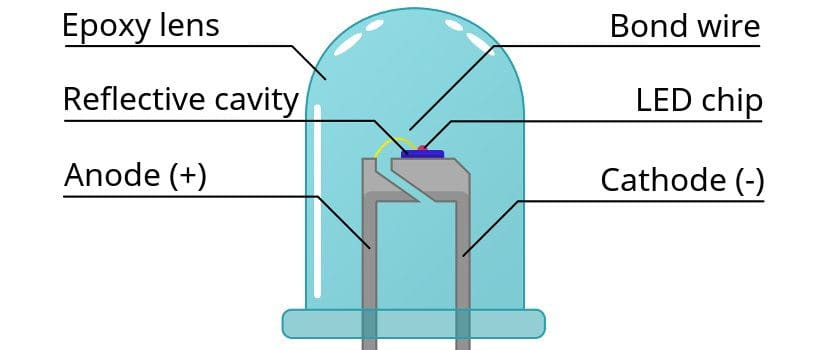
**Arduino Nano**

The Arduino Nano is a small, complete, and breadboard-friendly board based on the ATmega328P (Arduino Nano 3. x). It has more or less the same functionality of the Arduino Duemilanove, but in a different package. It lacks only a DC power jack, and works with a Mini-B USB cable instead of a standard one. It has 14 digital pins (D0-D13) and 8 analog input pins (D0-D7). Its small dimensions make it a good board for compact projects.

**Breadboard**

Breadboard is a solderless (plug-in) board for quick development of embedded systems projects, that makes it possible to connect different electronic parts to the development board using special wires without soldering. Solderless breadboards are great for building and testing new circuits because parts can be easily inserted and removed. They are completely re-usable.

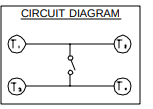
**LED (Light-Emitting Diode)**

** **

LED is a [semiconductor](https://en.wikipedia.org/wiki/Semiconductor) [light sourc](https://en.wikipedia.org/wiki/Light_source)e that emits light when [current](https://en.wikipedia.org/wiki/Electric_current) flows through it. [Electrons](https://en.wikipedia.org/wiki/Electron) in the semiconductor recombine with [electron holes](https://en.wikipedia.org/wiki/Electron_hole), releasing energy in the form of [photons](https://en.wikipedia.org/wiki/Photon). The color of the light (corresponding to the energy of the photons) is determined by the energy required for electrons to cross the [band gap](https://en.wikipedia.org/wiki/Band_gap) of the semiconductor. White light is obtained by using multiple semiconductors or a layer of light-emitting phosphor on the semiconductor device.

There are some advantages of LEDs over incandescent light bulbs. Unlike [incandescent bulbs](https://home.howstuffworks.com/light-bulb.htm), LEDs don't have filaments that burn out, they use less electricity, and they don't get very hot. LEDs last just as long as a standard transistor. The lifespan of an LED surpasses the short life of an incandescent bulb by thousands of hours.

**Button (push button)**

****

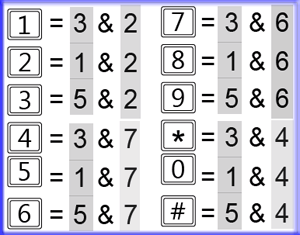
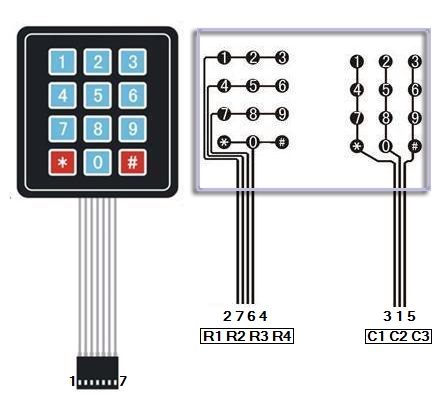
A push-button or simply button is a simple switch mechanism to control some aspect of a machine or a process. Buttons are typically made out of hard material, usually plastic or metal. These buttons usually contain a [spring](https://en.wikipedia.org/wiki/Spring_(device)) to return to their un-pushed state.

**Pull-up resistor**

Sometimes when having one pin configured as input, it is hard to tell if the state of the pin will be high (pulled to VCC) or low (pulled to ground). This phenomena is referred to as floating. To prevent this unknown state, a pull-up or pull-down resistor will ensure that the pin is in either a high or low state, while also using a low amount of current. [Pull-up resistors](http://en.wikipedia.org/wiki/Pull-up_resistor) are very common when using microcontrollers (MCUs) or any digital logic device.

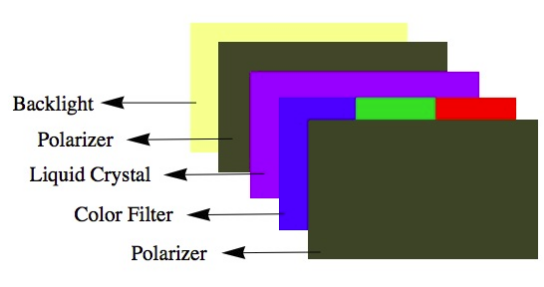
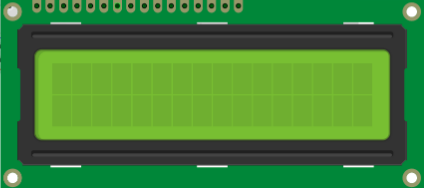
In this project the internal pull-up resistor of the microcontroller will be used.

**Keypad matrix (3x4)**

****

A keypad is one of the most commonly used input devices in microprocessor applications. In a standard keypad wired as an X-Y switch matrix, open switches connect a row to a column when pressed. A keypad with 12 keys (3x4) is wired as 3 columns by 4 rows.

**LCD (Liquid Crystal Display)**

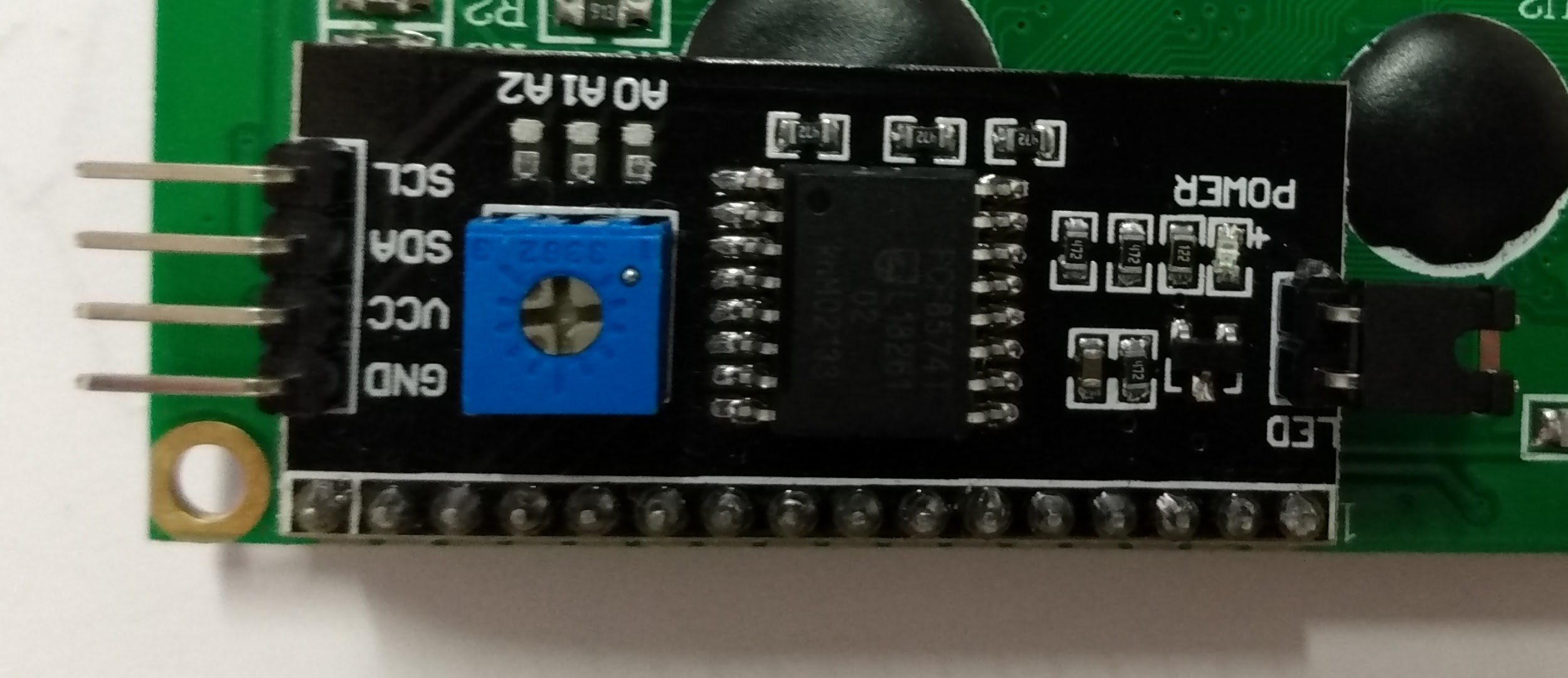
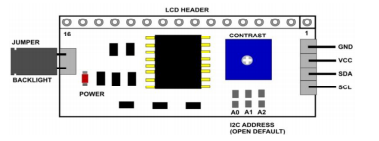
****

LCD (Liquid Crystal Display) is a type of flat panel display which uses liquid crystals in its primary form of operation. They work on the principle of blocking light.

Specifically, an LCD is made of two pieces of polarized glass (also called substrate) that contain a liquid crystal material between them. A backlight creates light that passes through the first substrate. At the same time, electrical currents cause the liquid crystal molecules to align to allow varying levels of light to pass through to the second substrate and create the colors and images on screen.

There are some advantages of LCDs. LCD displays are much thinner than cathode ray tube ([CRT](https://whatis.techtarget.com/definition/cathode-ray-tube-CRT)) technology and consume much less power than LED and gas-display displays.

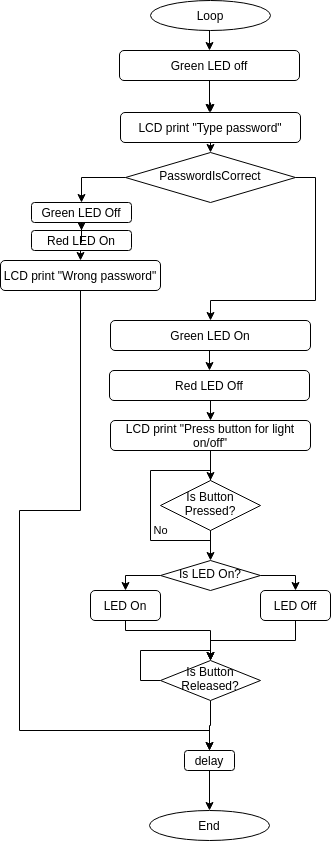
**I2C serial adaptor for LCD display**

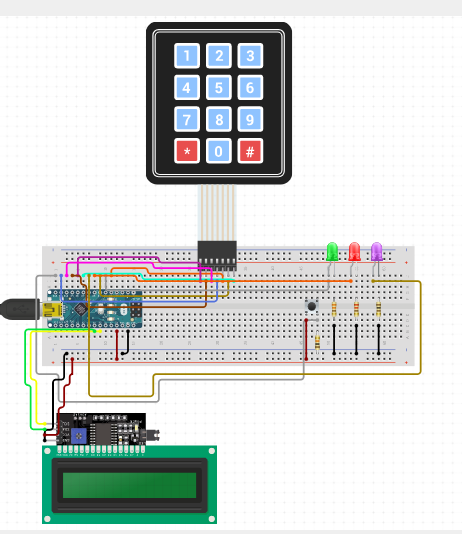
** **

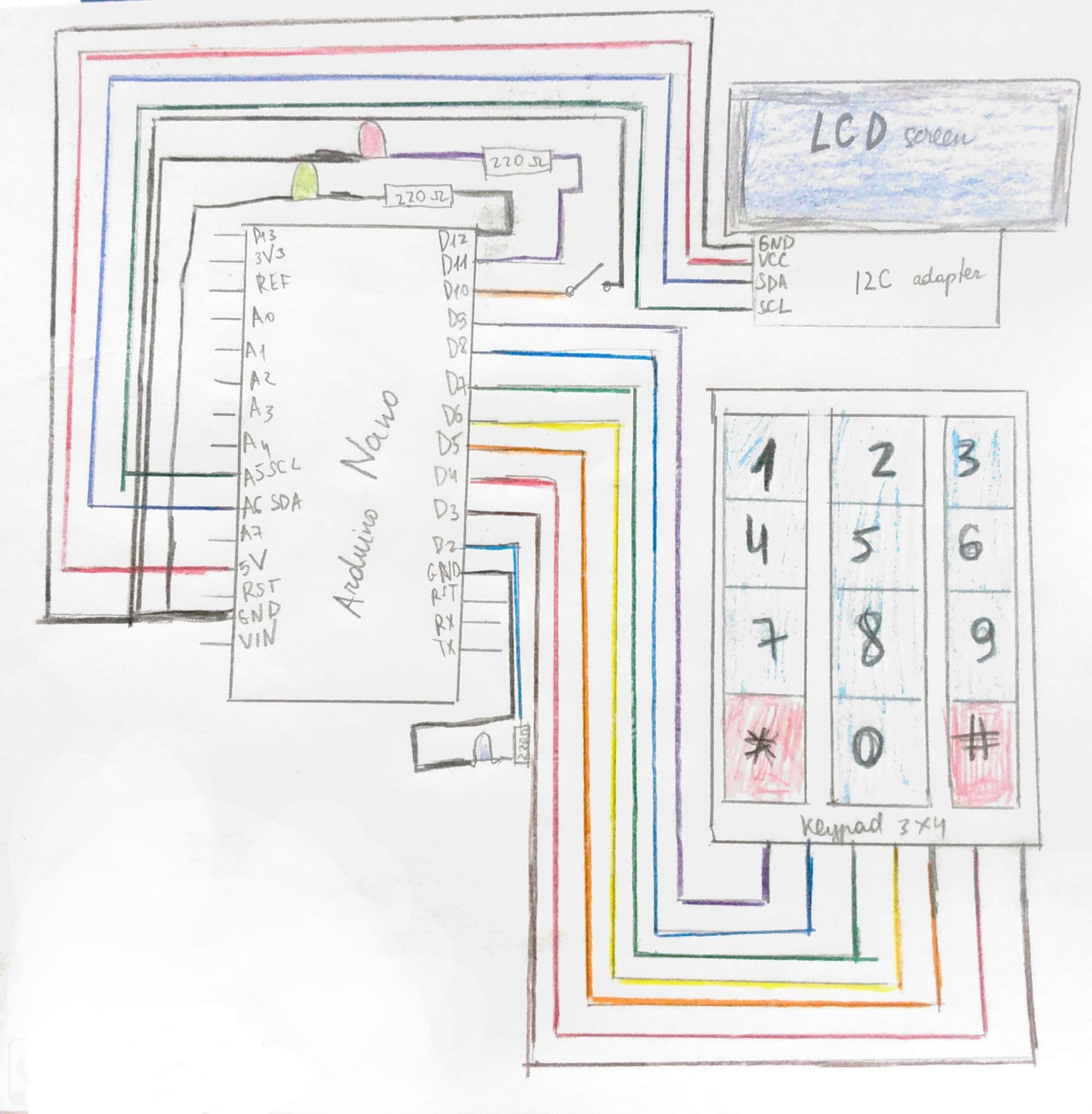
The LCD screen module for Arduino needs 16 pins to connect to Arduino board. This leaves less pins available for other components to connect to the development board. The **Converter Board I2C Serial Interface Module for LCD Display** makes it possible to control LCD using only 4 pins and allowing I2C interface communication.

|  |  |
| --- | --- |
| With LCD I2C adaptor | Without LCD I2C adaptor |
|  |  |

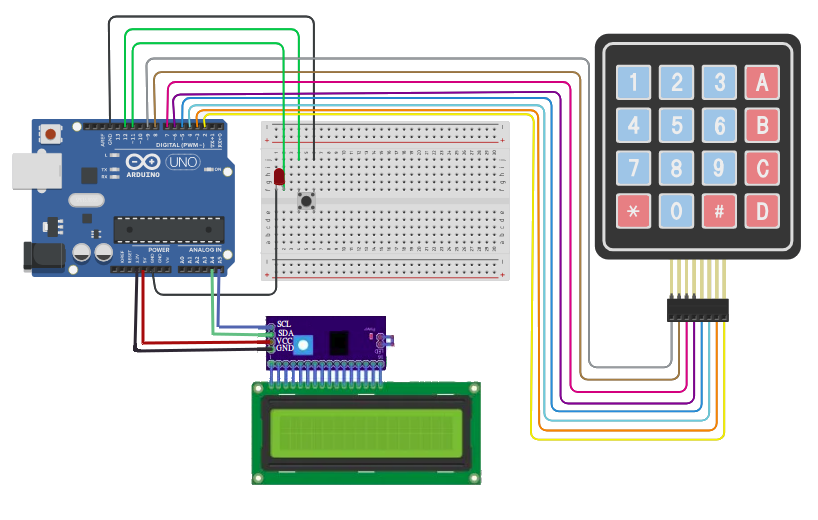
**Implementation**

** **

****

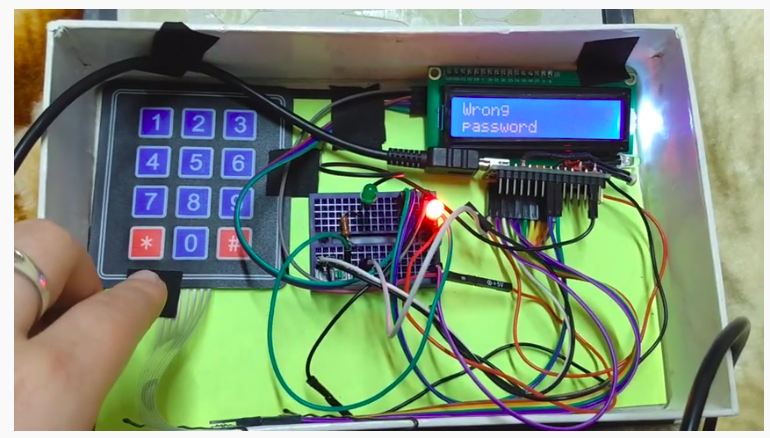
****

**Other possible implementation - Using Arduino Uno (not tested)**

****

**Proof**

[**https://www.youtube.com/watch?v=JPpAQtIyVl4**](https://www.youtube.com/watch?v=JPpAQtIyVl4)

[****](https://www.youtube.com/watch?v=JPpAQtIyVl4)

**Part 2 - serial communication terminal**

[**https://youtu.be/WOlXgPq90wU**](https://youtu.be/WOlXgPq90wU)

**Conclusions**

This project was a research and experiment on User Interactions using different hardware components and Arduino Nano development board. There were input components: button, 4x3 matrix keypad, serial monitor; and output components: serial monitor, LEDs and LCD screen.

All the components were connected in a system that can simulate the access and control of lights or electric current in a factory. Lights and electrical current are indispensable for a factory to function, and an unauthorized turn off can stop the activity and even cause losses of money. Therefore, using a keypad with input password helped authorizing user before turning on/off light and electrical current and preventing such problems. Also, two LEDs (red and green) were used for showing state - password correct/incorrect. In the case that LCD screen brokes, it is impossible for user to know the state of the system without these red and green LEDs. The button was used for turning on/off the light, but for bigger systems, users have multiple buttons, and a user can choose what button to use, but the authorization will be done once (no need to authorize for each button.

After planning and designing, this project was successfully implemented on real hardware components and Arduino Nano development board. Despite the fact that this is a small project, almost all the digital pins of the board were used, therefore using I2C adaptor for LCD screen was especially useful, because it used only 4 pins instead of 16.

In this project the output was displayed on multiple components (LCD screen, serial monitor), but using printf and standard output. This was useful because one printf statement could print at the same time on all devices and also because it can be easily modified to work with new devices.

User interactions are an important part of almost every project and by doing this laboratory it was possible to test various devices and make an integrated project that can simulate the control of lights and electrical current on a factory.

**Annex - Source code (1)**

//////////////////////////

// Button

/////////////////////////

//button.h

#define BUTTON\_PIN 10

#define BUTTON\_PRESSED 1

#define BUTTON\_RELEASED 0

void ButtonInit();

int IsButtonPressed();

int isButtonReleased();

////

void ButtonInit() {

pinMode(BUTTON\_PIN, INPUT\_PULLUP);

}

int IsButtonPressed() {

int buttonState = digitalRead(BUTTON\_PIN);

if(buttonState==BUTTON\_PRESSED) {

return 1;

}

//else

return 0;

}

int IsButtonReleased() {

int buttonState = digitalRead(BUTTON\_PIN);

if(buttonState==BUTTON\_RELEASED) {

return 1;

}

//else

return 0;

}

////////////////////////////////////////

// Led

////////////////////////////////////////

//led.h

// builtin pin is 13 - LED\_BUILTIN

#define LED\_PIN 2 // change

#define RED\_LED\_PIN 11 // change

#define GREEN\_LED\_PIN 12 // change

#define LED\_ON 1

#define LED\_OFF 0

void LED\_Init();

int Is\_LED\_On();

void LED\_On();

void LED\_Off();

////

void LED\_Init() {

// pinMode(LED\_PIN, INPUT\_PULLUP);

pinMode(LED\_PIN, OUTPUT);

pinMode(GREEN\_LED\_PIN, OUTPUT);

pinMode(RED\_LED\_PIN, OUTPUT);

}

int Is\_LED\_On() {

int LEDState = digitalRead(LED\_PIN);

if(LEDState==LED\_ON) {

return 1;

}

//else

return 0;

}

void LED\_On() {

digitalWrite(LED\_PIN, LED\_ON);

}

void Red\_LED\_On() {

digitalWrite(RED\_LED\_PIN, LED\_ON);

}

void Green\_LED\_On() {

digitalWrite(GREEN\_LED\_PIN, LED\_ON);

}

void LED\_Off() {

digitalWrite(LED\_PIN, LED\_OFF);

}

void Red\_LED\_Off() {

digitalWrite(RED\_LED\_PIN, LED\_OFF);

}

void Green\_LED\_Off() {

digitalWrite(GREEN\_LED\_PIN, LED\_OFF);

}

/////////////////////////////////////////

// Serial init

/////////////////////////////////////////

void SerialInit() {

Serial.begin(9600); // initializing serail monitor

}

/////////////////////////////////////////

//Keypad

/////////////////////////////////////////

#include <Keypad.h>

const byte Rows= 4;

const byte Cols= 3;

// key map as on the key pad:

char keymap[Rows][Cols]=

{

{'1', '2', '3'},

{'4', '5', '6'},

{'7', '8', '9'},

{'\*', '0', '#'}

};

byte rPins[Rows]= {3, 4, 5, 6}; //Rows 0 to 3

byte cPins[Cols]= {7,8,9}; //Columns 0 to 2

Keypad kpd = Keypad(makeKeymap(keymap), rPins, cPins, Rows, Cols);

char GetCharKeypad() {

char c = '+';

while(c=='+') {

char keypressed = kpd.getKey();

if (keypressed != NO\_KEY)

{ c = keypressed;

return c;

}

}

}

//////////////////////////////////

// LCD

/////////////////////////////////

#include <LiquidCrystal\_I2C.h>

#include <Wire.h>

#include <stdio.h>

//initialize the liquid crystal library

//the first parameter is the I2C address

//the second parameter is how many rows are on your screen

//the third parameter is how many columns are on your screen

LiquidCrystal\_I2C lcd(0x27, 16, 2);

void lcdOn() {

lcd.backlight();

}

void lcdOff() {

lcd.noBacklight();

}

void LcdInit() {

//initialize lcd screen

lcd.init();

// turn on the screen

lcdOn();

}

void PutCharLCD(char c) {

lcd.print(c);

}

///////////////////////////////

// Password

////////////////////////////////

char \* CORRECT\_PASSWORD = "1235";

int PasswordIsCorrect(char \* password) {

return strcmp(CORRECT\_PASSWORD, password);

}

char\* ReadPassword() {

char \* buf = (char \*) malloc (32);

int i = 0;

char c = '+';

while(c!='#') {

c = '+';

while(c=='+') {

char keypressed = kpd.getKey();

if (keypressed != NO\_KEY) {

c = keypressed;

printf("%c", c);

buf[i]= c;

i++;

}

}

}

return buf;

}

///////////////////////////////////

///////////////////////////////

// Functions for stdio

////////////////////////////////

int MyPutChar( char c, FILE \*t) {

Serial.write( c );

PutCharLCD(c);

}

char MyGetChar( FILE \*stream) {

return GetCharKeypad();

}

/////////////////////////////

// the setup function runs once when you press reset or power the board

void setup() {

LED\_Init();

ButtonInit();

SerialInit();

LcdInit();

fdevopen( &MyPutChar, 0);

fdevopen( &MyGetChar, 1);

lcd.setCursor(0,1);

printf("Hello!");

delay(1000);

}

// the loop function runs over and over again forever

void loop() {

Green\_LED\_Off();

lcd.clear();

lcd.setCursor(0,0);

printf("Type password #");

lcd.setCursor(0,1);

char \* password = (char \*) malloc(32);

password = ReadPassword();

Serial.print("Password:");

Serial.write(password);

lcd.clear();

if(PasswordIsCorrect(password)==1) {

lcd.setCursor(0,0);

printf("Correct");

lcd.setCursor(0,1);

printf("password");

Green\_LED\_On();

Red\_LED\_Off();

delay(800);

lcd.setCursor(0,0);

printf("Press button");

lcd.setCursor(0,1);

printf("for light On/Off\n");

delay(2000);

while(!IsButtonPressed());

if(Is\_LED\_On()) {

lcd.clear();

lcd.setCursor(0,0);

printf("LED off!\n" );

LED\_Off();

} else {

lcd.clear();

lcd.setCursor(0,0);

printf("LED on!\n");

LED\_On();

}

while(!IsButtonReleased());

} else {

lcd.setCursor(0,0);

printf("Wrong");

lcd.setCursor(0,1);

printf("password");

Green\_LED\_Off();

Red\_LED\_On();

}

delay(2000);

}

**Annex - Source code (2)**

#include <stdio.h>

int MyPutCh(char ch, FILE \*t) {

Serial.print(ch);

}

int MyGetCh(FILE \*t) {

while(!Serial.available());

char ch = Serial.read();

return ch;

}

void SerialInit() {

Serial.begin(9600);

fdevopen( &MyPutCh, &MyGetCh);

}

void setup() {

SerialInit();

}

void loop() {

char \* text = (char \*) malloc (64); // 8 chars - sizeof char = 8 bits

printf("Type something\n");

scanf("%s", text);

printf("You typed:%s\n", text);

int number;

printf("Type integer\n");

scanf("%d", &number);

printf("You typed:%d\n", number);

delay(500);

}