Islamic University of Gaza
Faculty of Engineering
Department of Computer Engineering

Embedded Systems Lab 2023-ECOM 4022
Eng.Ghydaa T.Mohaisen
Eng.Baraa Alastal

Lab 1: Introduction to the Embedded Lab

Outlines:

Introduction to the Embedded systems

An embedded system is a computer system designed to perform one or a few dedicated functions, often with real-time computing constraints. It is embedded as part of a complete device often including hardware and mechanical parts. Embedded systems control many devices in common use today,Examples include Automobiles, Mobile phones, Medical equipment, Industrial control systems, Personal Digital Assistants (PDAs), GPS Receivers, Video Game Consoles, And Digital Cameras.

The key characteristic of an embedded system is that it is designed to do one or a few specific tasks and it is integrated with the device it is controlling. Unlike general-purpose computers, such as personal computers or servers, embedded systems have a specific function, which makes them more efficient and less expensive.

Creating embedded systems means working with very low and hardware-specific stuff, you should be very aware of the hardware itself, and the components connected to it to make it works well.Then you should know that working with these systems means that you should work with a very time-dependent system, these systems need you to be very precise as each millisecond will make your system behave differently.

The main components of an embedded system

1. **Microcontroller:** A microcontroller is a small computer on a single integrated circuit. It is the heart of the embedded system and is responsible for executing the instructions of the program. It is also responsible for controlling the input and output devices.
2. **Memory:** Embedded systems have both read-only memory (ROM) and random access memory (RAM). ROM is used to store the program code and data that cannot be changed during operation, while RAM is used for data storage that can be read and written.
3. **Input/Output (I/O) interfaces:** to communicate with the outside world, Examples include serial and parallel ports, USB ports, and Ethernet ports.
4. Timers/Counters:to precise timing, and this is achieved using timers or counters. Timers can be used to generate regular interrupt signals, while counters can be used to keep track of events.
5. **Peripherals:**such as sensors, actuators, displays, and communication interfaces, which are used to interface with the outside world.
6. Power supply: can be a battery, a DC power supply, or an AC-to-DC converter.
7. **Software:** The software is the set of instructions that tell the microcontroller how to perform its task. It is usually written in a high-level language such as C or assembly.

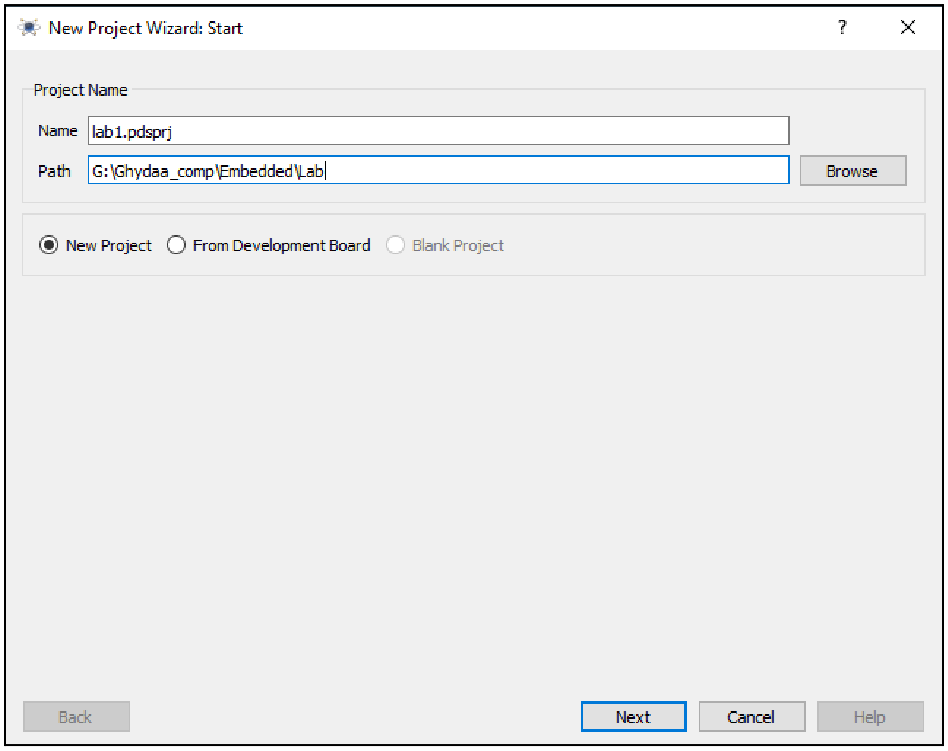
LPC1114 - Microcontrollers

- Based on the ARM Cortex-M0 processor core
- Up to 32 KB of flash memory for program storage
- Up to 8 KB of SRAM for data storage
- Has some peripherals like UART, I2C, and SPI communication interfaces
- ADC and DAC for analog-to-digital and digital-to-analog conversions
- GPIO for general-purpose input/output

Datasheet link:
https://www.keil.com/dd/docs/datashts/nxp/lpc11xx/lpc111x_lpc11cxx_um.pdf

Installing Proteus

We will use Proteus as our main IDE, for writing code, and for simulation. install Proteus 8 Professional and make sure that its version is 8.9, other versions may make some problems. So to create a new project, follow these steps:

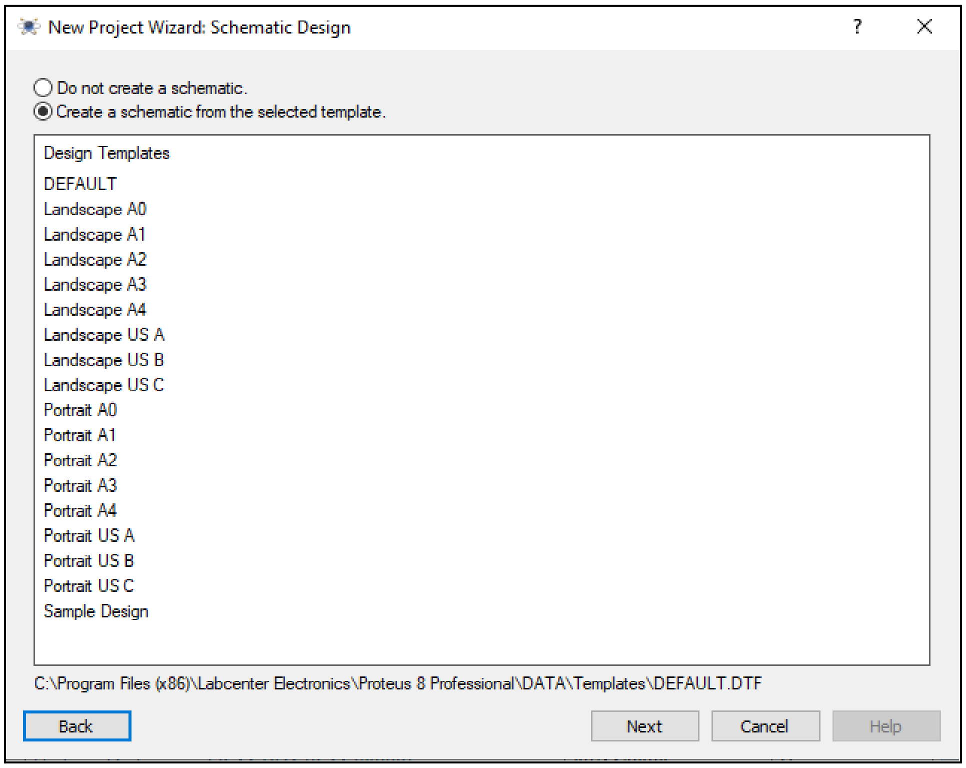

New Project Wizard: Start

Project Name

Name: lab1.pdsprj

Path: G:\Ghydaa_comp\Embedded\Lab

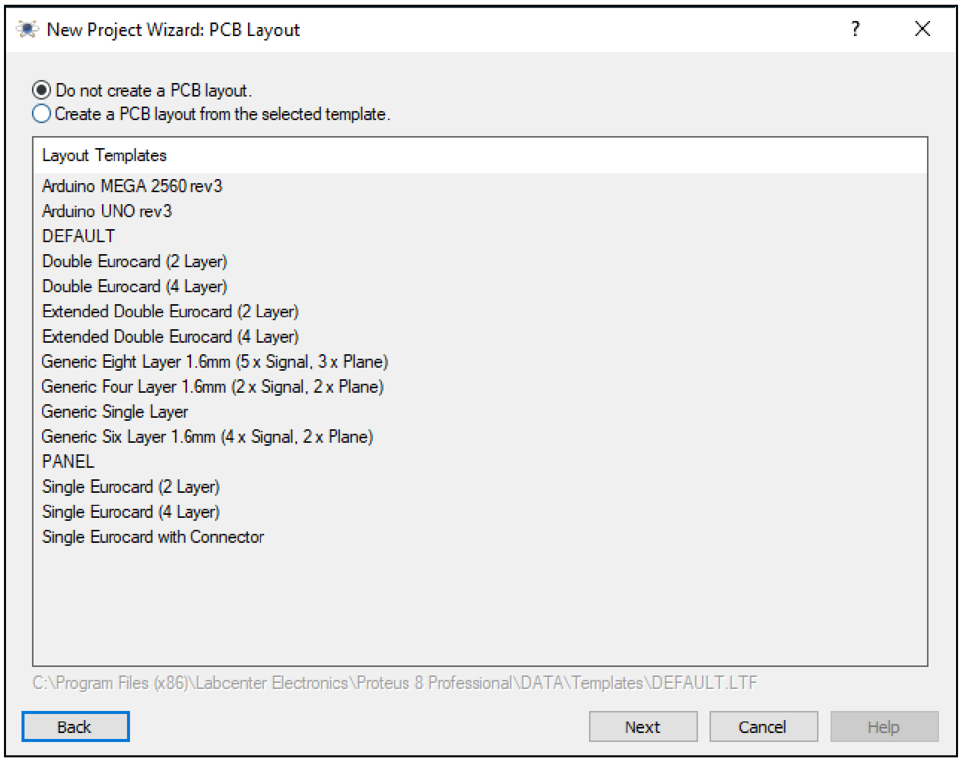
[✓ New Project] From Development Board Blank Project



( ) Do not create a schematic.
(●) Create a schematic from the selected template.

Design Templates
DEFAULT
Landscape A0
Landscape A1
Landscape A2
Landscape A3
Landscape A4
Landscape US A
Landscape US B
Landscape US C
Portrait A0
Portrait A1
Portrait A2
Portrait A3
Portrait A4
Portrait US A
Portrait US B
Portrait US C
Sample Design

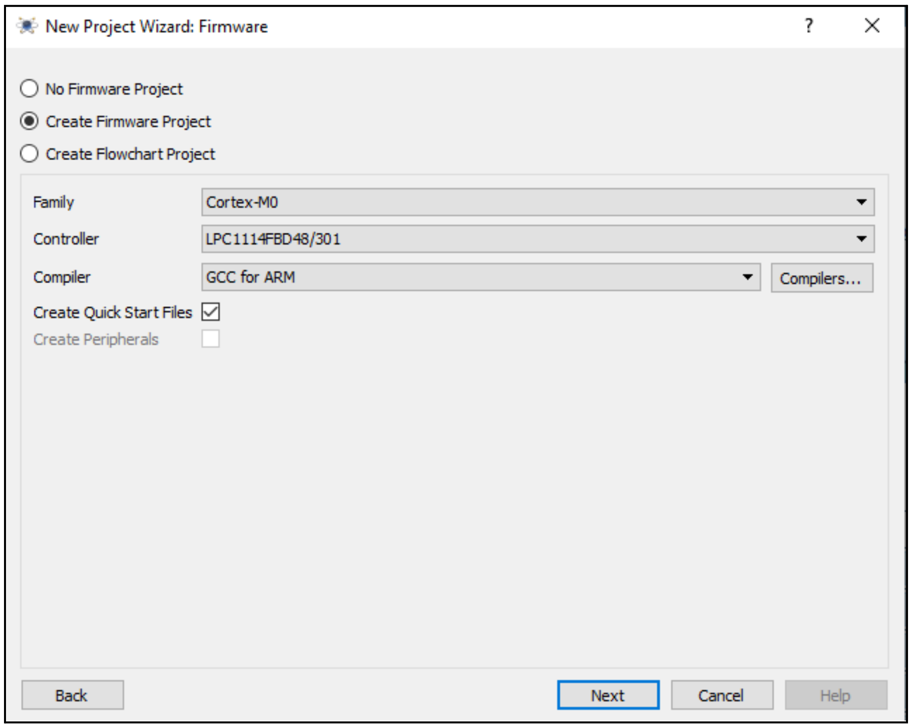
C:\Program Files (x86)\Labcenter Electronics\Proteus 8 Professional\DATA\Templates\DEFAULT.DTF



(●) Do not create a PCB layout.
( ) Create a PCB layout from the selected template.

Layout Templates
Arduino MEGA 2560 rev3
Arduino UNO rev3
DEFAULT
Double Eurocard (2 Layer)
Double Eurocard (4 Layer)
Extended Double Eurocard (2 Layer)
Extended Double Eurocard (4 Layer)
Generic Eight Layer 1.6mm (5 x Signal, 3 x Plane)
Generic Four Layer 1.6mm (2 x Signal, 2 x Plane)
Generic Single Layer
Generic Six Layer 1.6mm (4 x Signal, 2 x Plane)
PANEL
Single Eurocard (2 Layer)
Single Eurocard (4 Layer)
Single Eurocard with Connector

C:\Program Files (x86)\Labcenter Electronics\Proteus 8 Professional\DATA\Templates\DEFAULT.LTF


New Project Wizard: Firmware

○ No Firmware Project
● Create Firmware Project
○ Create Flowchart Project

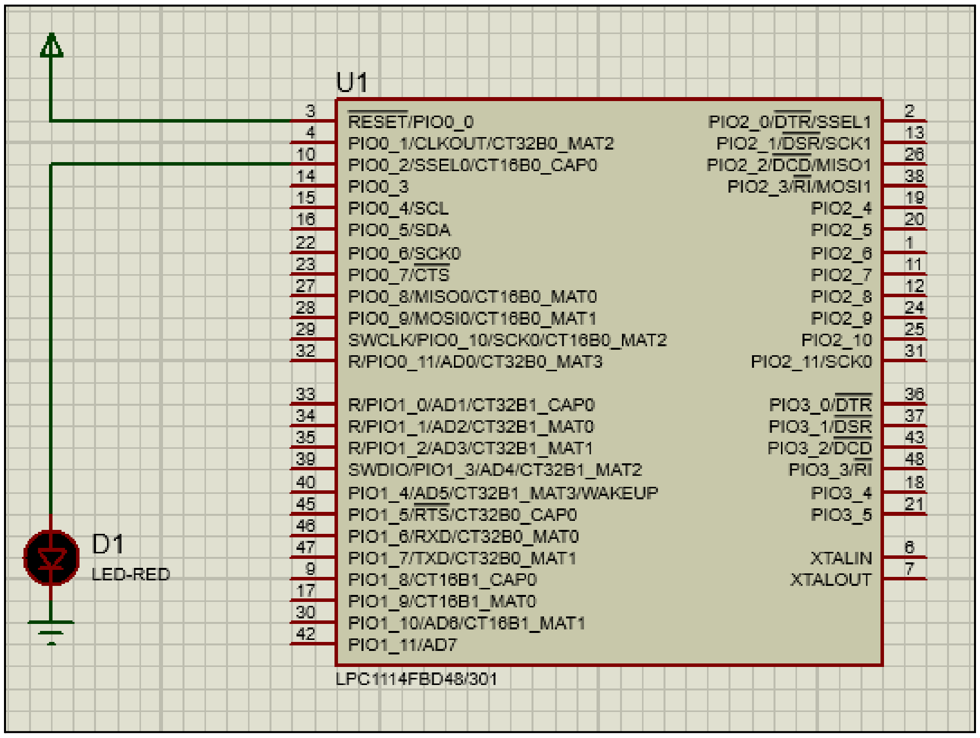
Family: Cortex-M0
Controller: LPC1114FBD48/301
Compiler: GCC for ARM
Create Quick Start Files: ✓
Create Peripherals: ☐

Note: The GCC for ARM compiler might not be installed, click on "Compilers…" and install it.

Examples

Example 01:

In this example, we will write a small code that reads the GPIO ports, we will read PORT0 data, and we will write back on it, so the purpose of this example is to flip the status of the LED repeatedly.



```
#define GPIO0DIR (*((volatile unsigned long *)0x50008000))
#define GPIO0DATA (*((volatile unsigned long *)0x50003FFC))
int main () {
 //setup phase
 GPIO0DIR |= 0b100;
 int i = 0;
 while (1) {
 //looping phase
  GPIO0DATA^=0b100;
    for(i=0;i<100000;i++);
 }
}
```

NOTES:

1. **GPIO0DIR** is a register that is used to configure the direction of the GPIO pins on the CPU. It is used to set the pins as inputs or outputs, and is typically used in embedded systems.if the value of a bit is 0, the corresponding pin is configured as an input. If the value of a bit is 1, the corresponding pin is configured as an output.

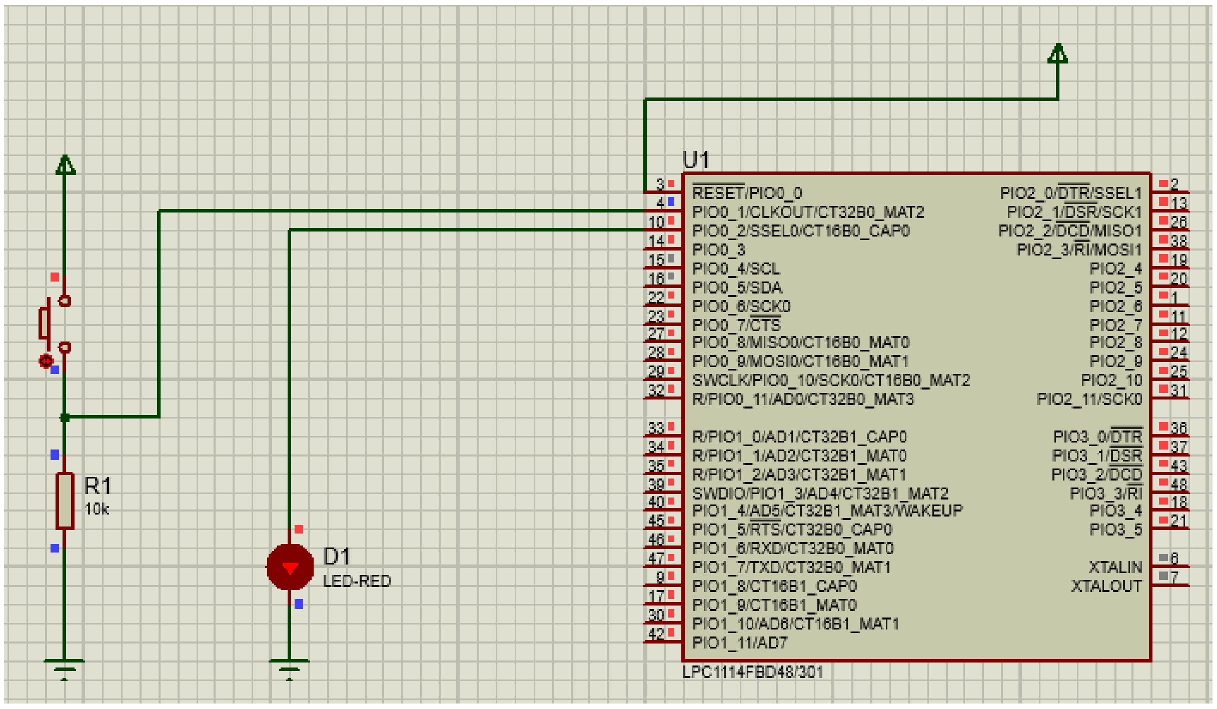
2. **GPIO0DATA** is a register in microcontrollers that stands for General Purpose Input/Output 0 Data. It is used to read the value of input pins or to write the value to output pins, depending on the configuration of the corresponding GPIO0DIR register.

if the value of the corresponding bit in the GPIO0DATA register is 1, the level of the signal at the pin will be logic 1. If the value of the corresponding bit in the GPIO0DATA register is 0, the level of the signal at the pin will be logic 0.

3. **Volatile keyword** the compiler is instructed not to perform any optimizations that may change the value of the variable unexpectedly. For example, the compiler may normally cache the value of a variable in a register, rather than accessing the variable in memory each time it is used. However, if a variable is declared as volatile, the compiler will always access the variable in memory, so that changes made by other sources are immediately visible.

Example 02:

Flip the status of the LED in case of you pushed a button.



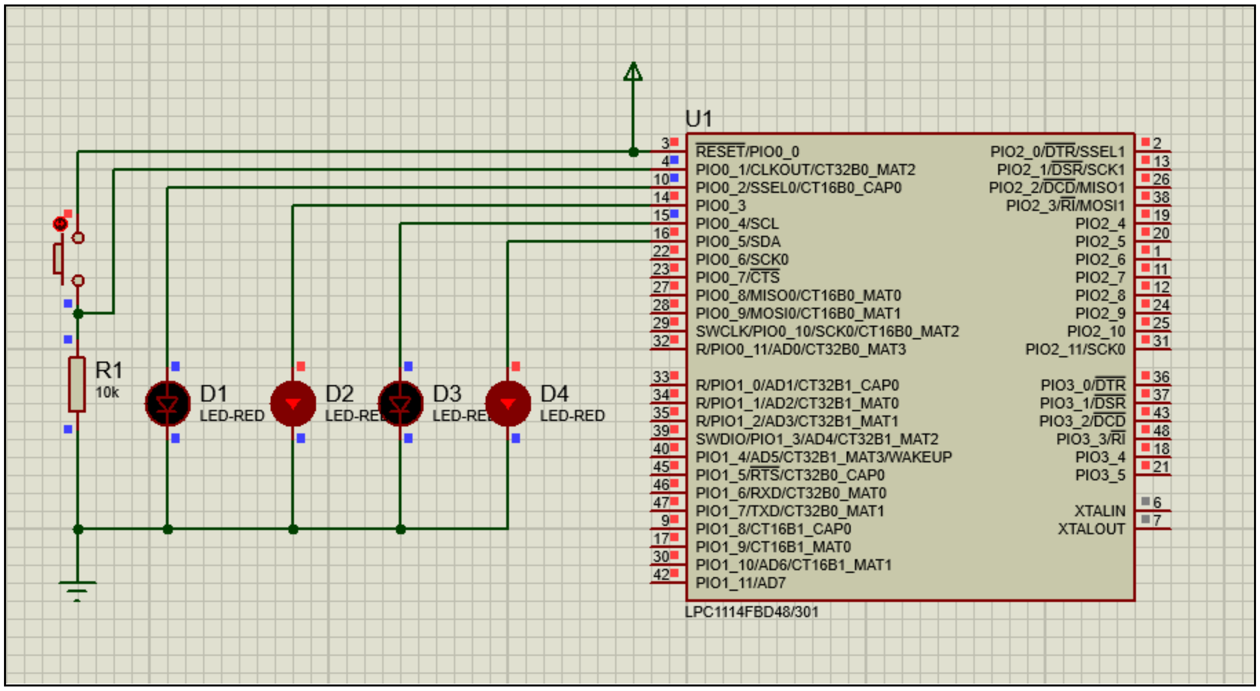
```
#define GPIO0DIR (*((volatile unsigned long *)0x50008000))
#define GPIO0DATA (*((volatile unsigned long *)0x50003FFC))
int main () {
 //setup phase
 GPIO0DIR =0b100;
 while (1) {
 //looping phase
 if (GPIO0DATA & 0b10) {
 GPIO0DATA ^=0b100;
 while(GPIO0DATA & 0b10);
 }}}
```

A pull-down resistor

When a switch or other input device is open (i.e., not activated), the input signal is effectively floating and can be influenced by various environmental factors such as electromagnetic interference (EMI) or noise. A pull-down resistor provides a low-impedance path to the ground reference voltage, ensuring that the input signal is pulled to a logic low level when the switch is open or when no input signal is present.

Example 03:

Flip the status of LEDs in case of you pushed a button "each 2 LEDs of the 4 will be in the same state "



```
#define GPIO0DIR (*((volatile unsigned long *)0x50008000))
#define GPIO0DATA (*((volatile unsigned long *)0x50003FFC))
int main () {
 //setup phase
 GPIO0DIR |= 0x3c;
 GPIO0DATA = 0x28;
 while (1) {
 //looping phase
 if (GPIO0DATA & 0x2) {
 GPIO0DATA ^= 0x3c;
 while ((GPIO0DATA & 0x2));
 }
 }
}
```

Lab Tasks

Task #1: In this task, you should edit the code above, to make a design that takes two buttons as input, if this first one clicked you should shine 4 LEDs (the four LEDs should be connected to PORT1), then if you clicked the second one you should turn all the LEDs off.

Task #2: Using the your code and proteus project in Task #1, edit the code so that just one LED from the four LEDs will shine on the begging, and if the first button is clicked you should shift the shining LED to left, and if the second button is clicked you should shift the shining LED to the right, (Turn off the LED and turn on the sibling LED), implement this as rotation.