Embedded system
must respond quickly to sequences and combinations of events
must perform multiple separate activities simultaneously
must handle problems without crashing

example : closed-loop control systems
maintain set point
specialized functions within larger systems

microcontroller based es
+ beetter performance/efficienty, cost-effective, fewer components
- cost, size, weight, battery capacity, heating issues

IoT(Internet of Things)
: everything should be able to communicate with anything

Price and number of embedded systems have inverse proportional relationship.

Mbed : platform for Cortex-based application and systems development

RTOS : Real time core, semaphores, Locks/Mutexes
Development Tools ; Mbed studio, Arm Mbed Online complier, Mbed CLI : commnad line tool
security : TLS(Transport Layer security), SPM(Secure Partition Manager)
Code testing : Greentea

Mbed platform : software libraries, hardware development kit

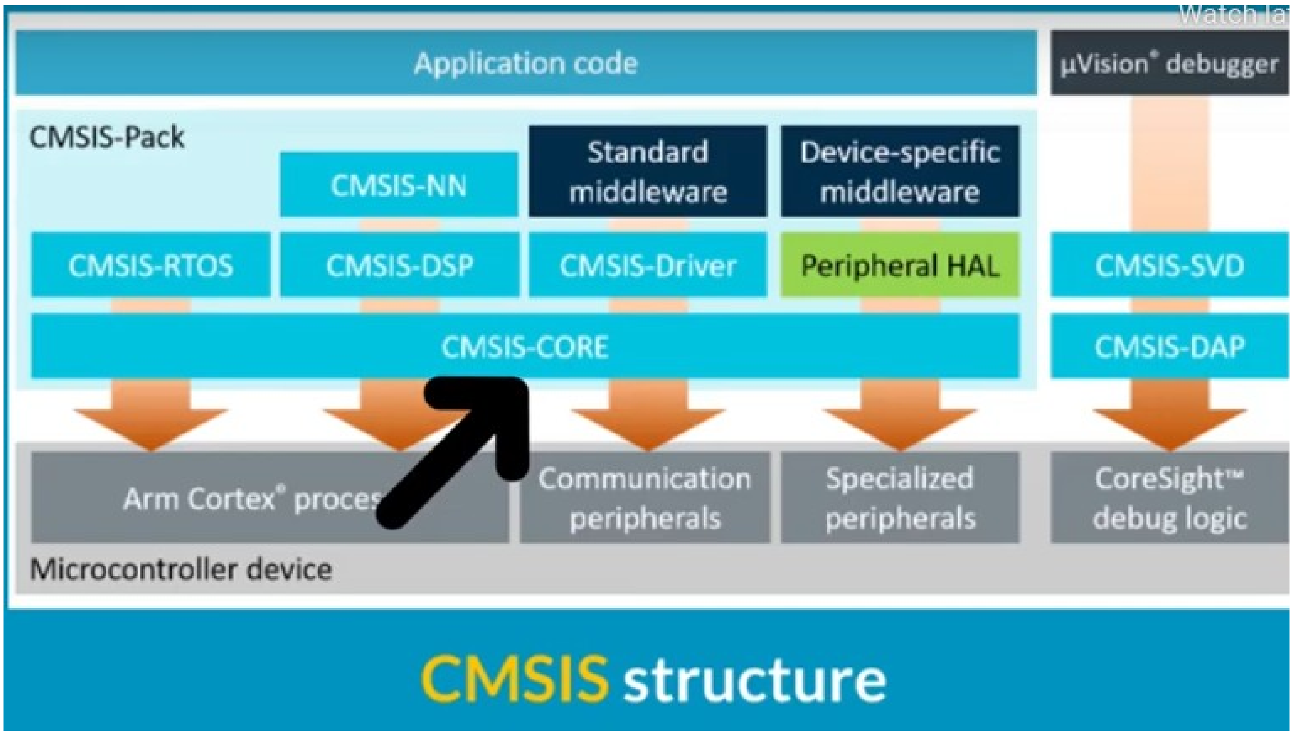
High-level programming : shorter development time, portabilitiy, code is easier to read, maintain and reuse
but less optimized code, longer translation time

low-level programming

+ better optimized code, memory efficiency, less translation time, programmer can talk directly to hardware
- less portability, code is harder to read, maintain and reuse, increased development time

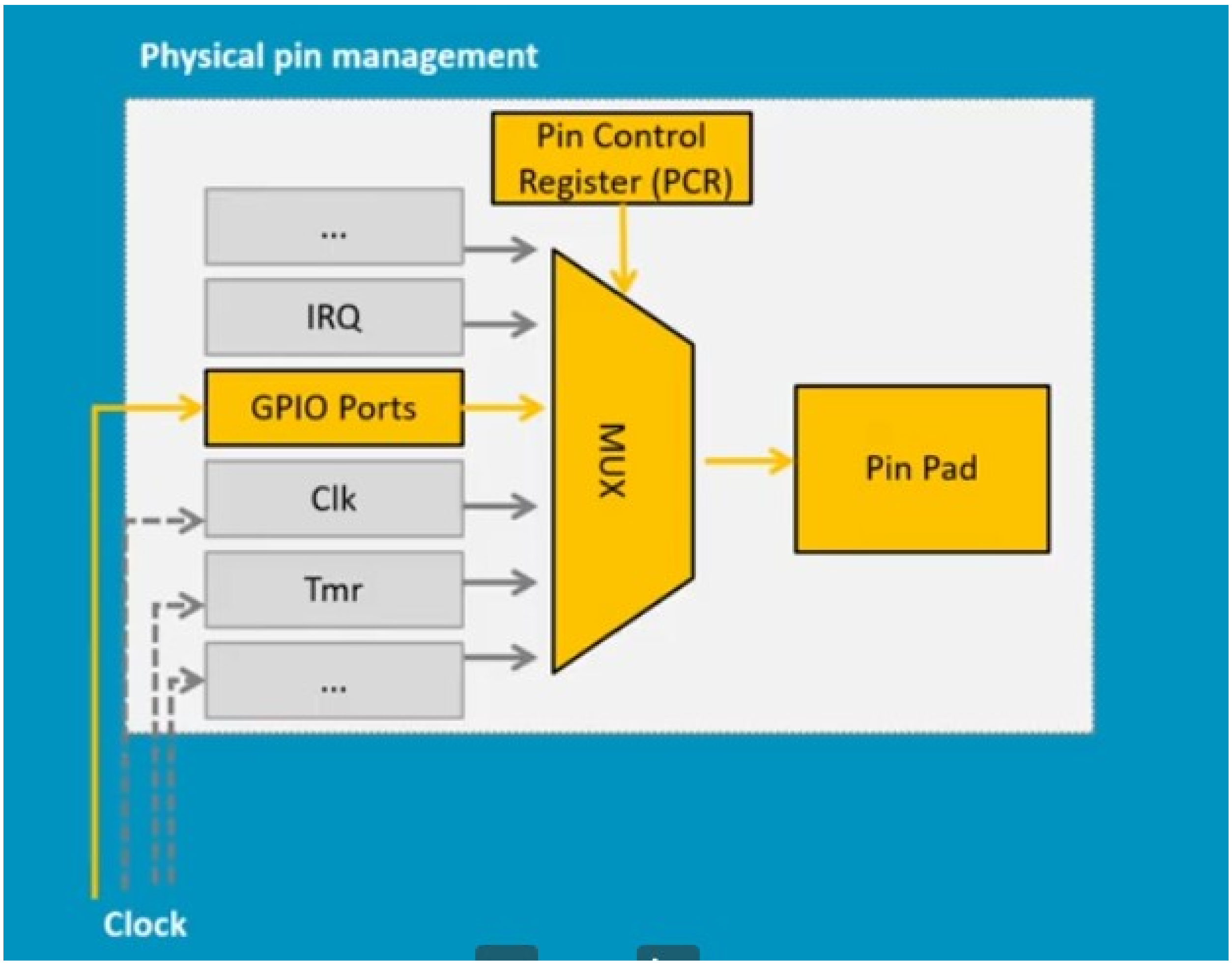
Cortex Microcontroller Software Interface Standard(CMSIS)
: standardizations,
access to special registers
functions to access special instructions

+ portable code, better compatibility with third party software, better code density, smaller memory footprint



CMSIS -RTOS : provides common API for real time operating systems

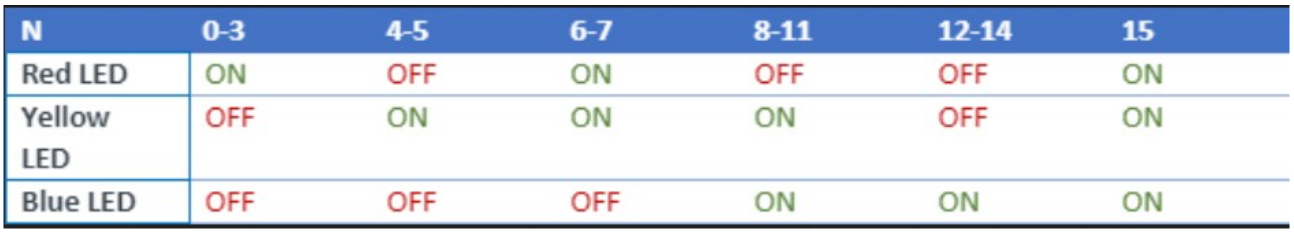
```
Docker
Docker enables access to Mbed Simulator image file easily.



MODULE 3
1. Voltages and Logic Values

Logic circuits : can cope with some noise, can cope with some signal
degradation. Not only describing the data with 0 and 1 but with various values
in recognizable signals.

2. GPIO (General Purpose Input Output)
configurable for a range of signals

switches control power supply,
only two voltage levels : 0 and +vdd
+vdd = 1.8V
use Ohm’s law to find resistor value
R = (VDD-VLED)/ILED
R = resistor value
VDD = source of current
VLED  = voltage required by LED
ILED = current through resistor


Floating : situation when there is no input. The program cannot decide if the
switch should be high or low

3. GPIO design and microcontroller
```



```
physical pins are assigned dynamically to different functions.
GPIO should be set separately to each physical pin.
```

The multiplexer is managed by a pin control register, or PCR

Lab session

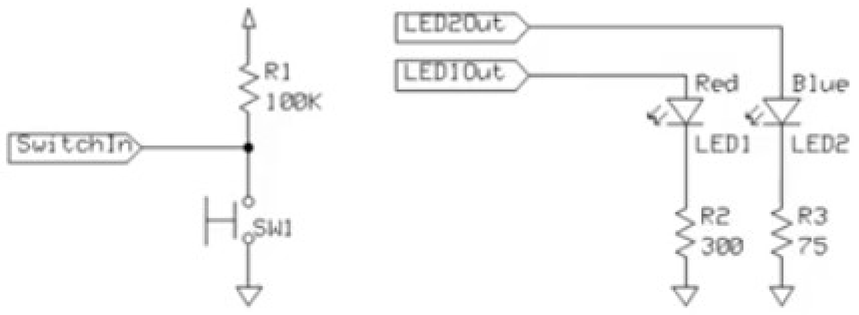


N	0-3	4-5	6-7	8-11	12-14	15
Red LED	ON	OFF	ON	OFF	OFF	ON
Yellow LED	OFF	ON	ON	ON	OFF	ON
Blue LED	OFF	OFF	OFF	ON	ON	ON

LED reference table

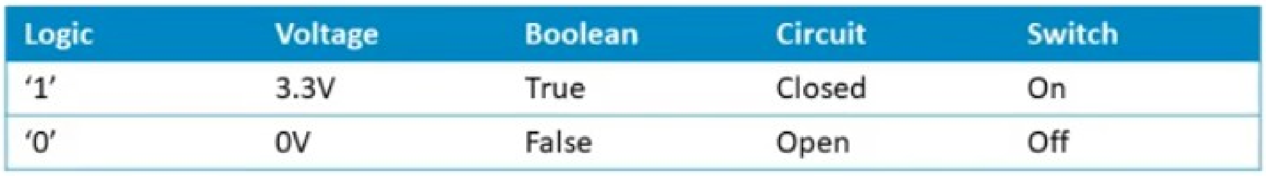
Goal : light either LED1 or LED2 based on switch SW1 position

input : switch
output : LEDs



Circuit
diagram for our lab, connecting one switch to two LEDs
to our Cortex processor

GPIO = general-purpose input and output
input : program can determine if input signal is 1 or 0
output : program can set output to 1 or 0



Logic	Voltage	Boolean	Circuit	Switch
'1'	3.3V	True	Closed	On
'0'	0V	False	Open	Off

GPIOs can only provide two voltage levels. If more than two is required, the analog input and output or the trick of a pulse width modulation, must be used.

Pull up and Pull-down resistors
The purpose of pull-up and pull-down resistors is to ensure that there is a default or "known" value on the input pin to prevent an unknown state called "floating"

floating : situation where there is no input, the program cannot decide whether the pin will be high or low

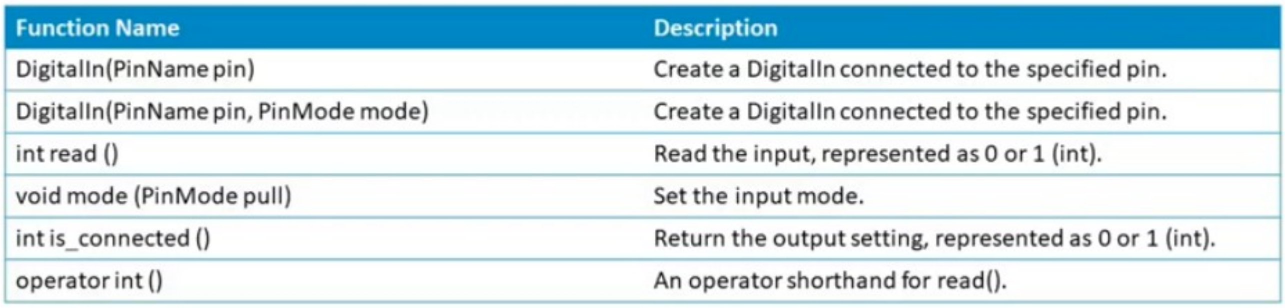
Mbed API : provides a number of drivers that provide access to general purpose microcontroller hardware

key API
- DigitalIn : to read the value of a digital input pin where the logic level is either 1 or 0
- DigitalOut : to configure and control a digital output pin by setting the pin to a logic level of 0 or 1

any number of Arm Mbed pins can be used as a DigitalIn or DigitalOut

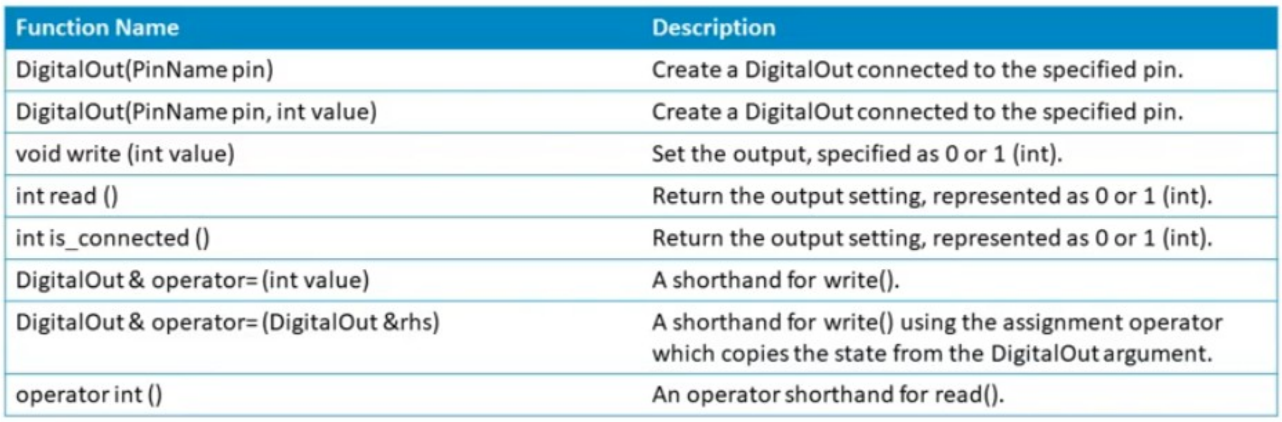
```
DigitalIn mybutton(Input Pin);
DigitalOutLed_out(OutputPin);

int main(){
if(mybutton)
Led_out = 1;
}
```

how a digital input pin called "mybutton" and a digital output pin called "Led_out" to turn on an LED if a button is pressed



Function Name	Description
DigitalIn(PinName pin)	Create a DigitalIn connected to the specified pin.
DigitalIn(PinName pin, PinMode mode)	Create a DigitalIn connected to the specified pin.
int read ()	Read the input, represented as 0 or 1 (int).
void mode (PinMode pull)	Set the input mode.
int is_connected ()	Return the output setting, represented as 0 or 1 (int).
operator int ()	An operator shorthand for read().

DigitalIn class reference



Function Name	Description
DigitalOut(PinName pin)	Create a DigitalOut connected to the specified pin.
DigitalOut(PinName pin, int value)	Create a DigitalOut connected to the specified pin.
void write (int value)	Set the output, specified as 0 or 1 (int).
int read ()	Return the output setting, represented as 0 or 1 (int).
int is_connected ()	Return the output setting, represented as 0 or 1 (int).
DigitalOut & operator= (int value)	A shorthand for write().
DigitalOut & operator= (DigitalOut &rhs)	A shorthand for write() using the assignment operator which copies the state from the DigitalOut argument.
operator int ()	An operator shorthand for read().

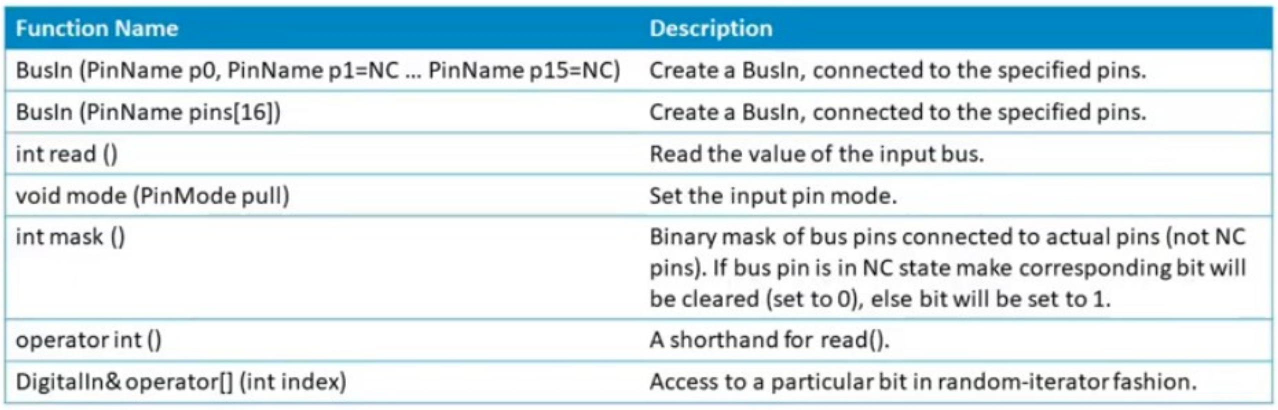
DigitalOut class reference

- BusIn : allows to combine a number of DigitalIn pins to read them at once
  : useful for checking multiple inputs together as a single interface instead of individual pins
- BusOut : allows to combine a number of DigitalOut pins to write to them at once

: useful for writing to multiple pins together as a single interface instead of individual pins

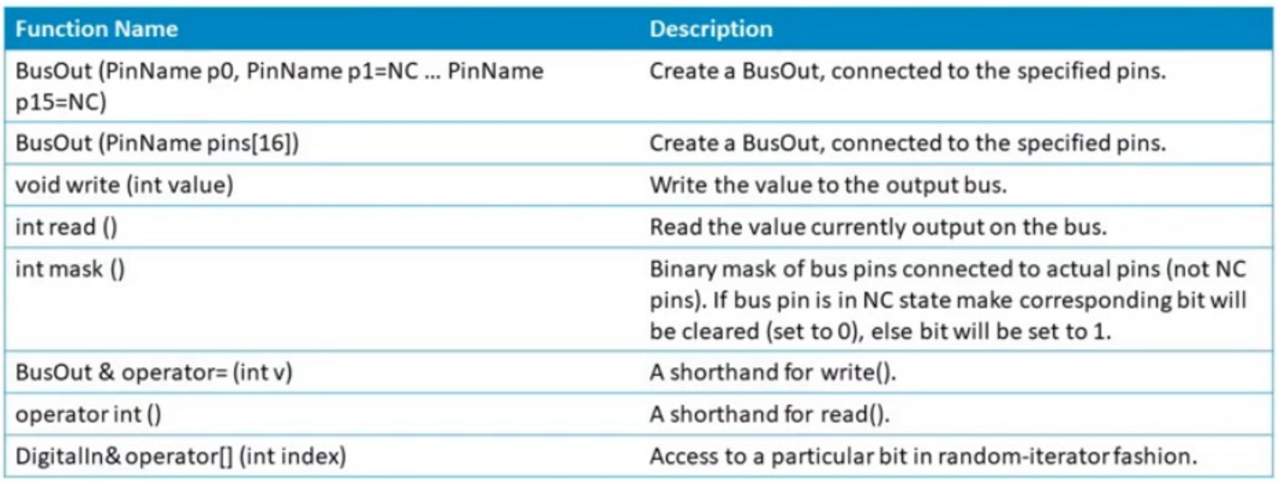
```
#include “mbed.h”

BusOut myleds(LED1, LED2, LED3, LED4);

int main(){
while(1){
for(int I =0; i<16; i++){
myleds = I;
wait(0.25);
}}}
```

any number of Arm Mbed pins can be used as a BusIn or BusOut



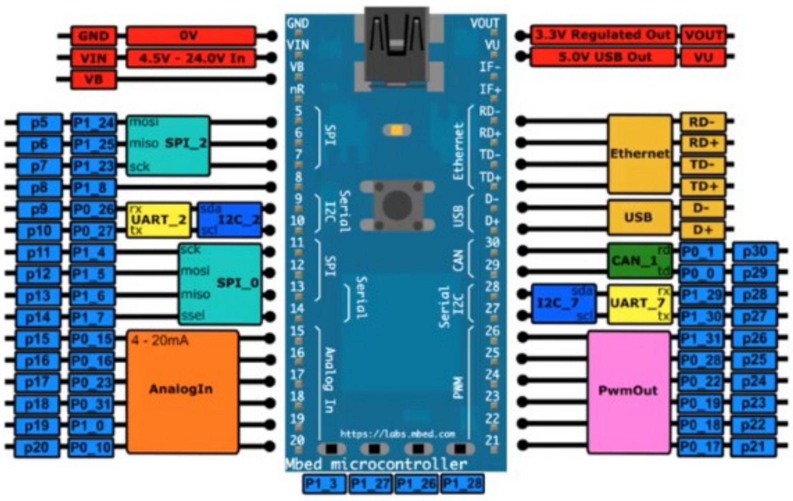
Function Name	Description
BusIn (PinName p0, PinName p1=NC ... PinName p15=NC)	Create a BusIn, connected to the specified pins.
BusIn (PinName pins[16])	Create a BusIn, connected to the specified pins.
int read ()	Read the value of the input bus.
void mode (PinMode pull)	Set the input pin mode.
int mask ()	Binary mask of bus pins connected to actual pins (not NC pins). If bus pin is in NC state make corresponding bit will be cleared (set to 0), else bit will be set to 1.
operator int ()	A shorthand for read().
DigitalIn& operator[] (int index)	Access to a particular bit in random-iterator fashion.

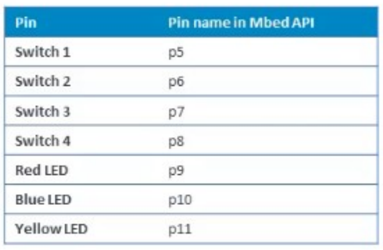
BusIn class reference



Function Name	Description
BusOut (PinName p0, PinName p1=NC ... PinName p15=NC)	Create a BusOut, connected to the specified pins.
BusOut (PinName pins[16])	Create a BusOut, connected to the specified pins.
void write (int value)	Write the value to the output bus.
int read ()	Read the value currently output on the bus.
int mask ()	Binary mask of bus pins connected to actual pins (not NC pins). If bus pin is in NC state make corresponding bit will be cleared (set to 0), else bit will be set to 1.
BusOut & operator= (int v)	A shorthand for write().
operator int ()	A shorthand for read().
DigitalIn& operator[] (int index)	Access to a particular bit in random-iterator fashion.

BusOut class reference





Pin	Pin name in Mbed API
Switch 1	p5
Switch 2	p6
Switch 3	p7
Switch 4	p8
Red LED	p9
Blue LED	p10
Yellow LED	p11

diagram mapping out the design of the simulated microcontroller.