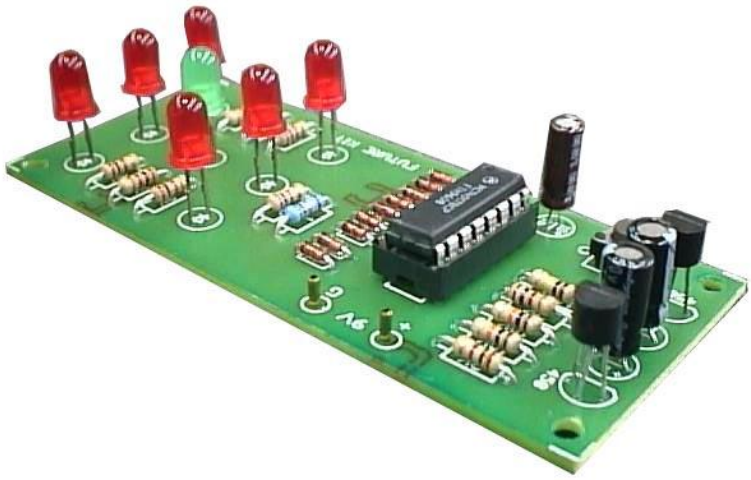
Electronic Dice Using Microchip's PIC12F629



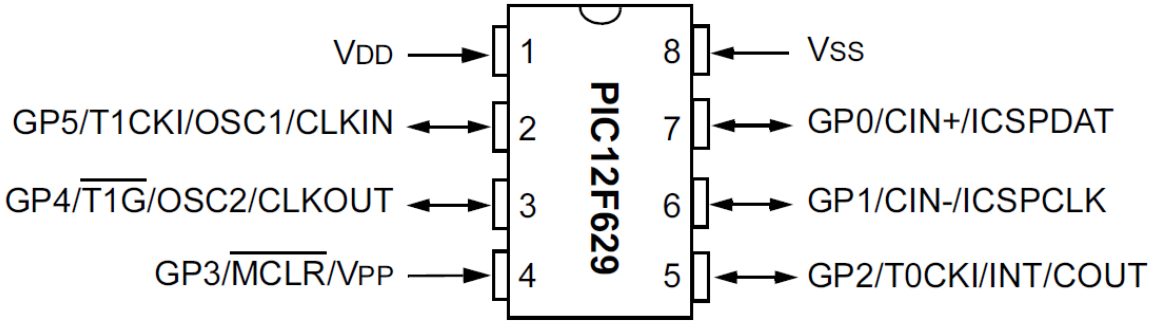
Submitted by,

1. Sambhav R Jain 107108103
2. Vangapandu Srivijay 107108078
3. Kommanaboina Pramod 107108060
4. Arul Goutham 107108044
5. Naseef Thalib 107108017

Why use an 8 pin microcontroller?

- The low cost 8-pin PIC12F629, released by Microchip has a re-programmable FLASH memory, making it ideal for use in educational projects.
- Its low cost also makes it far more accessible for class work
- Simple circuit construction with only 8 pins to configure
- Easy PCB layout
- 5 or 6 input/output pins, including one analogue input

Pin Diagram:



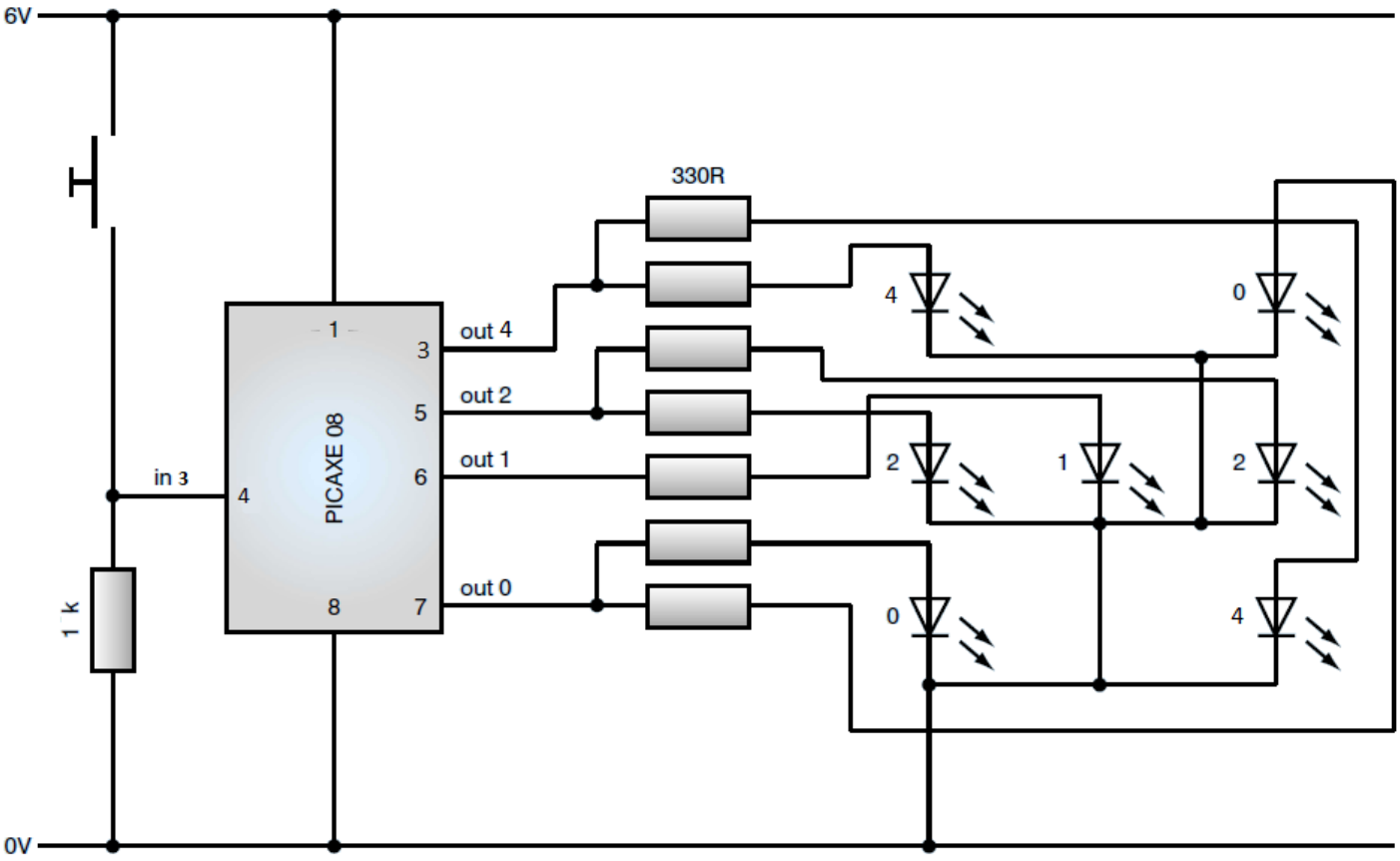
Caution:

Microchip do not use the same 'leg' numbers as the 'input/output pin' numbers and so make sure that there is a difference between the input/output pin (I/O pin) number and the physical leg number!

Description of the Electronic Dice:

This project demonstrates how the microcontroller can be used to switch output LEDs on and off to make an electronic dice for a game. Naturally this circuit could also be adapted for many other projects – e.g. a bicycle safety light which flashes high intensity LEDs on and off in patterns.

From the circuit diagram below, it can be seen that the I/O pin 3 (leg 4) is configured as an input and has a switch connected. The other pins are used as outputs and are connected to the LEDs which are laid out in the dot pattern of a traditional dice. Note that the diagonal pairs of LEDs are driven from the same I/O pin as they also light at the same time.



Thus symmetry in a dice is used to reduce the number of output pins required to control these LEDs from 7, down to only 4. This enables us to use the cheap and apt 8-pin PICAXE 08 (PIC12F629).

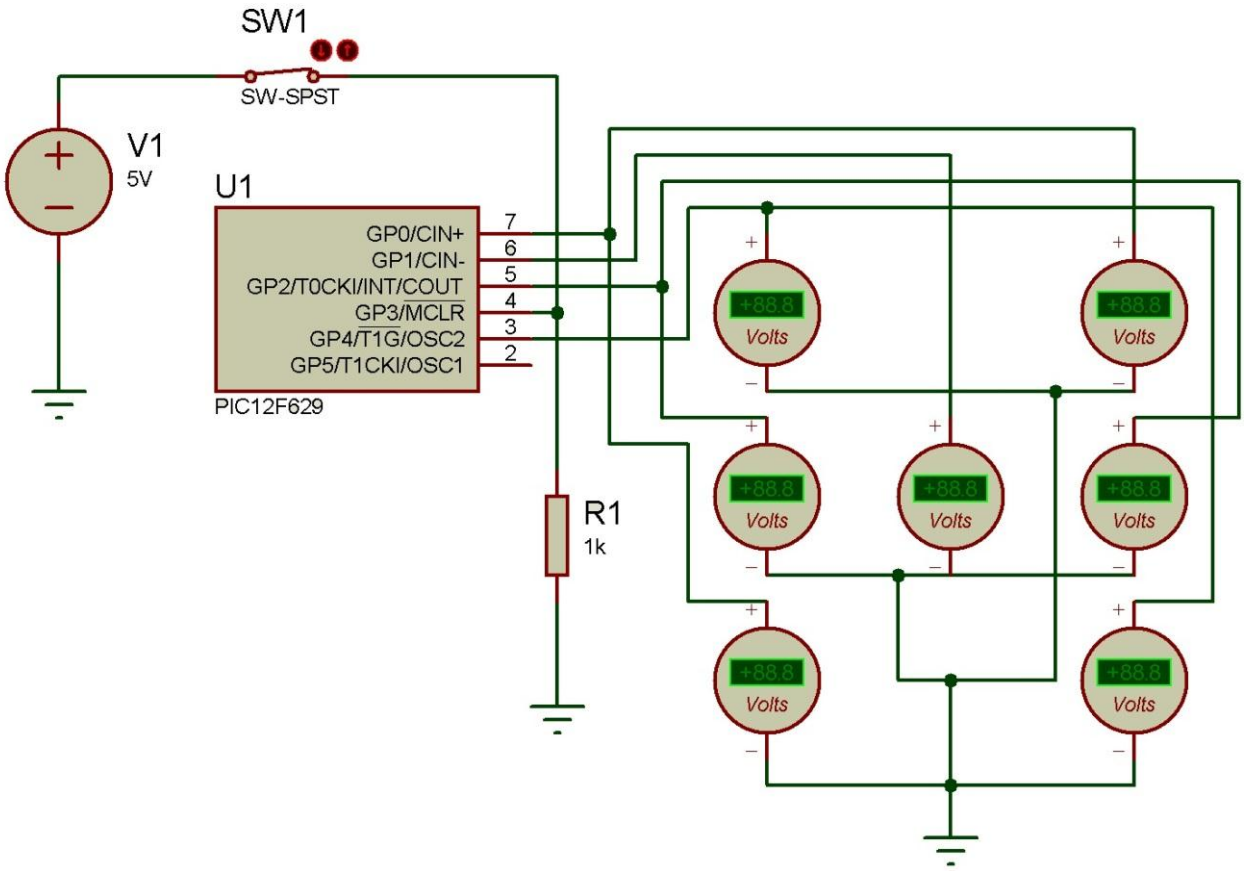
Assembly code:

```
;*****************************************************
; File Name:    source.asm
; Description: Electronic Dice
; Author:       Sambhav R Jain
; Target:       Microchip PIC 12F629 Microcontroller
; Compiler:     Microchip Assembler (MPASM)
; IDE:          Microchip MPLAB IDE v8.63
;*****************************************************

list p=PIC12F629          ; selects the device
#include <p12F629.inc>    ; include header file for selected PIC

errorlevel -302           ; no warnings (registers not in bank 0)

radix hex                 ; hexadecimal

;--------------------
;Configuration Bits
;--------------------

; internal oscillator selected, watch-dog timer OFF, power-up
timer ON, no ext reset, brown-out detect OFF, no program memory
code and data memory code protection

__CONFIG _INTRC_OSC_NOCLKOUT & _WDT_OFF & _PWRTE_ON & _MCLRE_OFF
& _BODEN_OFF & _CP_OFF & _CPD_OFF

;-----------------------------------------------------------------

          org 0x0000
start     movlw 0x00        ; disable all interrupts
          movwf INTCON
          bcf STATUS,RP0    ; select bank 0
          clrf GPIO         ; init. GPIO
          movlw 0x07
          movwf CMCON       ; GP<2:0> are digital IO pins
          bsf STATUS,RP0    ; select bank 1
          movlw b'00001000' ; GP0,GP1,GP2,GP4 - configure as o/p
          movwf TRISIO      ; GP3 - configure as i/p
          bcf STATUS,RP0    ; select bank 0
repeat    movlw 0x02        ; show dots for number 1
          movwf GPIO
```

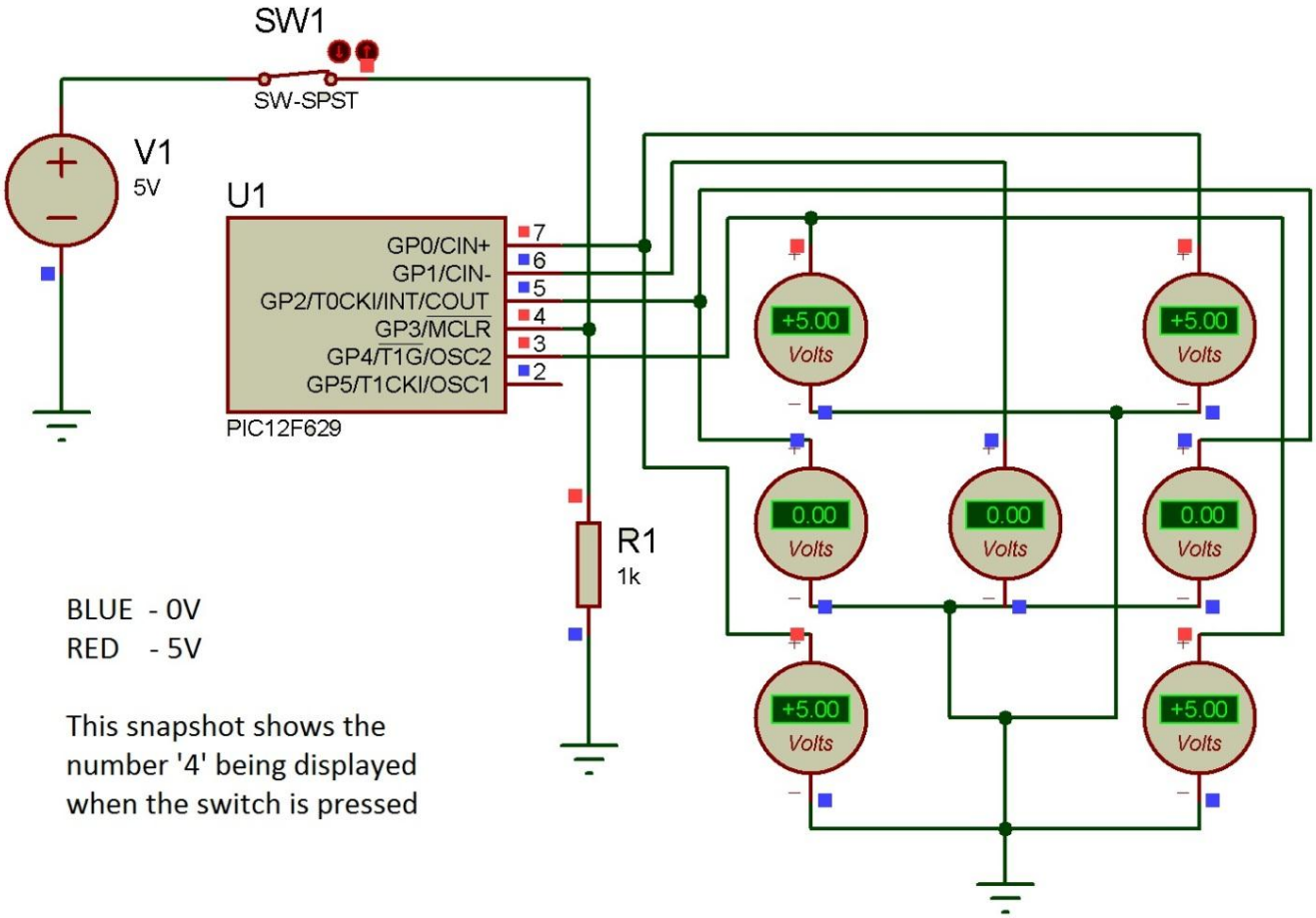
```
            call delay
            movlw 0x10 ; show dots for number 2
            movwf GPIO
            call delay
            movlw 0x12 ; show dots for number 3
            movwf GPIO
            call delay
            movlw 0x11 ; show dots for number 4
            movwf GPIO
            call delay
            movlw 0x13 ; show dots for number 5
            movwf GPIO
            call delay
            movlw 0x15 ; show dots for number 6
            movwf GPIO
            call delay
            goto repeat
delay       btfss GPIO,GP3 ; check if input switch is pressed, if
                           yes, then stay within the loop, else
                           return
            return
            goto delay

            end
;*****************************************************
```

Simulation Results:

The assembly code is compiled using MPLAB SIM and the generated HEX code is imported to Proteus Model that is shown below. In order to view the changes at the output, dc voltmeters are used in the place of LEDs.



A pull-down resistor is used at the input pin 4 (GP3), to pull the voltage down, when in input is LOW. The next figure shows one of the states of the dice, when the switch is pressed.



Now, the code is burnt onto the PIC12F629, using the PICSTART Plus Programmer. Since PIC12F629 uses Flash memory, the code can be erased and re-written onto it.

Results and Discussion:

1. Specifically, electronic dices are already in use in all casinos (say in Las Vegas)
2. This kind of a circuit does not limit to the application in a dice, but can be used in many fields where high intensity LED or LASER is required to be controlled in a definite pattern (e.g. remote communication in railways or airways)
3. Field of applications also include *cryptography* (key generation) and *random noise generation* (study performance of systems subjected to noise)
4. Future expansion applies to arts and creative pattern forming, used in art galleries