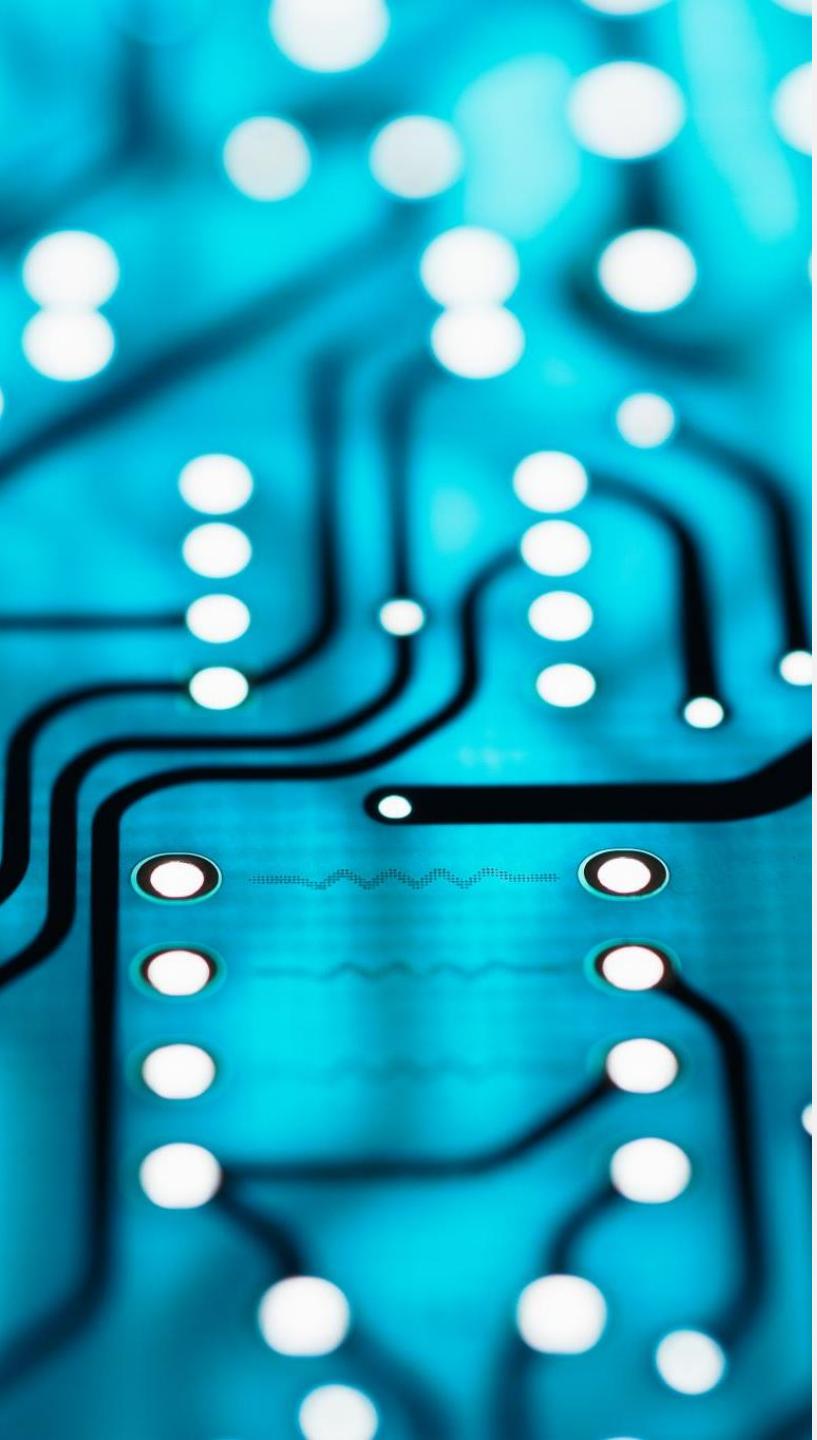


Embedded Systems (EPM)

Lecture (4) Summary

Timers Functions:

- **mills(): measure the time in ms since the board is started**
- **micros(): measure the time in us since the board is started**
- **delay(): stops the program for the specified period in ms**
- **delayMicroseconds(): stops the program for the specified period in us**

Making Manual Timer

This program turn on, off this LED every 250 mS

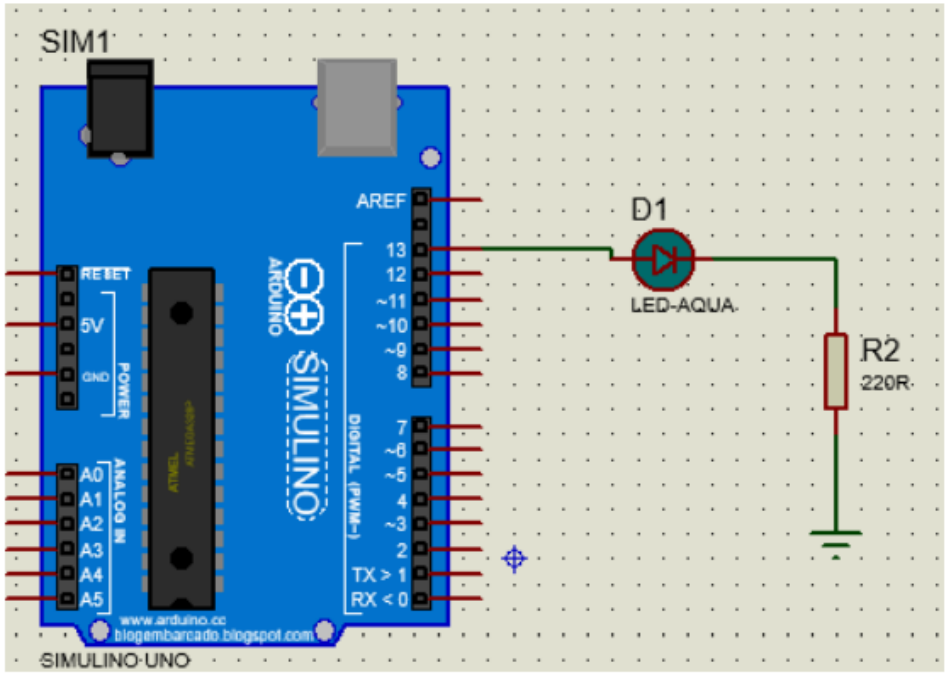
```
#define LED 13

void flash() {
      static boolean output = HIGH;
      digitalWrite(LED, output);
      output = !output;
}
int oldTime = 0;
void setup() {
      pinMode(LED, OUTPUT);
}
void loop() {
      int time = millis();
      if((time-oldTime)>250)
      {
            flash();
            oldTime = time;
      }
}
```

<-- this function toggles the LED condition every 250 ms. we declare the Variable (output) as STATIC because if its boolean only every time the function runs the Variable(output) will be HIGH (=1) neither the value on LED

Notes:

1. This is a manual non real-time timer.
2. Timer interval may exceed 250 ms.

This Program is not Precise 100% because if there's many codes after the condition in VOID LOOP() , it will delay more than 250 mS until it return to the condition again



MsTimer2 Library functions:

- **set(interval, callbackfunction): Set the real-time timer interval in ms, and sets the callback function name**

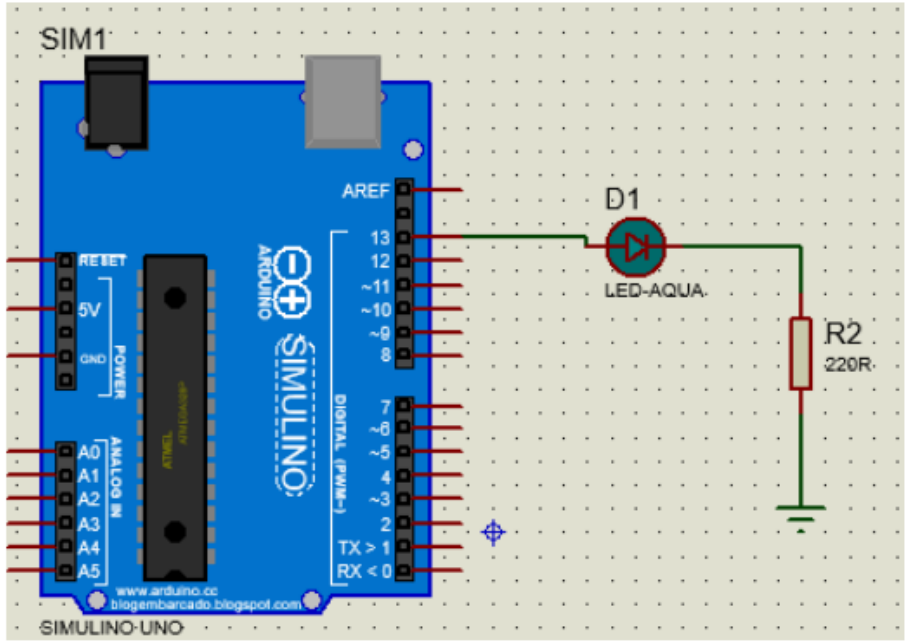
- **start(): Starts the timer**

- **stop(): Stops the timer**

Using MsTimer2 Library

```
#include <MsTimer2.h>

#define LED 13

void flash() {
      static boolean output = HIGH;
      digitalWrite(LED, output);
      output = !output;
}

void setup() {
      pinMode(LED, OUTPUT);
      MsTimer2::set(500, flash);
      MsTimer2::start();
}

void loop() {
}
```

This Program Solves the problem of the previous Program as its a real time Timer



Notes:

1. This is a real-time timer.

2. Timer interval exactly equals 250 ms.

Reading Analog Signal:

ATMega328 A/D Converter:

- Read 6 analog inputs
- Analog range : 0→5V / 0→3.3V depending on the power signal (VCC)
- Resolution: 10 bit
- Digital range : 0→1023
- 1 bit change = 5V/1024 = 0.0049V
- Use analogReference(type) function to change the range bellow the maximum

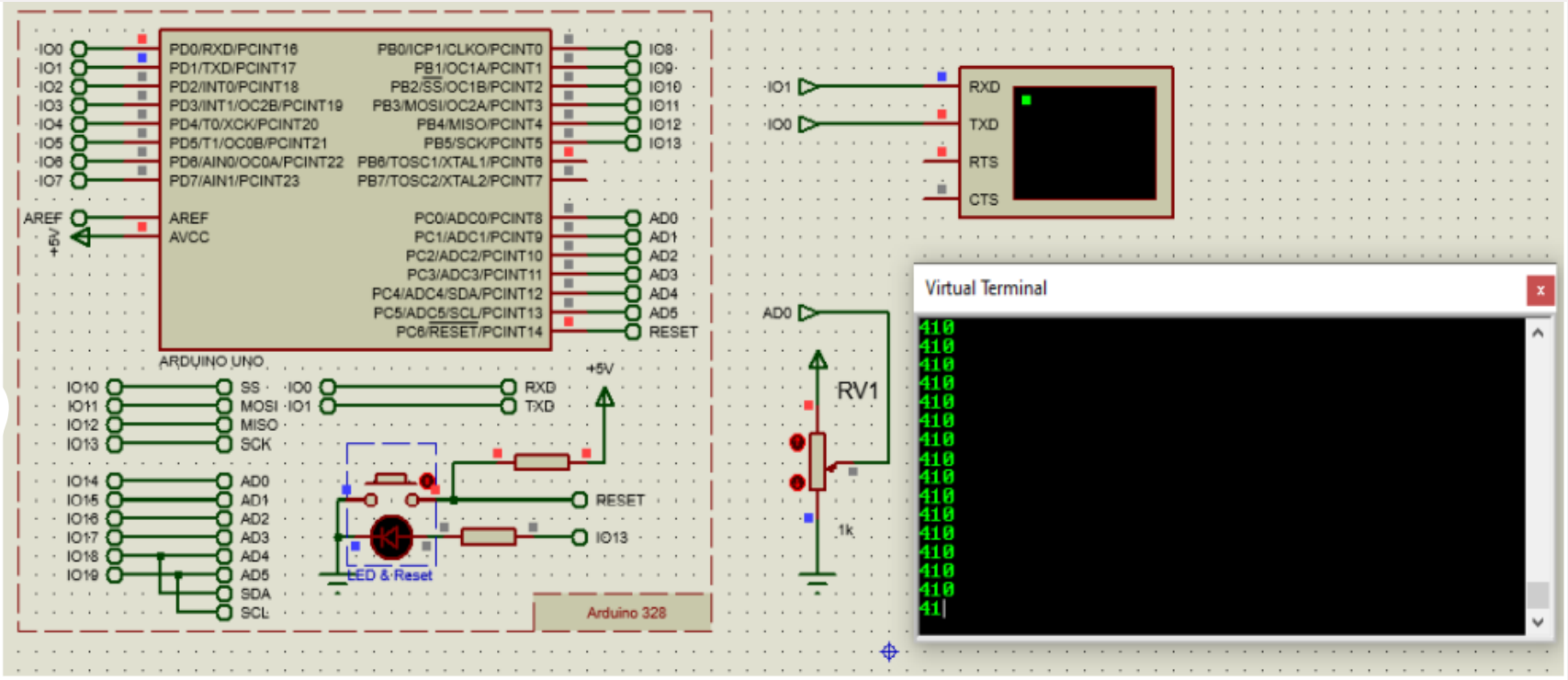
Changing A/D Input Voltage Range

- Using **analogReference(type)** function

- Type can take:
  - DEFAULT: 5V or 3.3V based on Board Type
  - INTERNAL: 1.1V for UNO Boards, 2.56 for Mega Boards
  - INTERNAL1V1: 1.1V for Mega Boards
  - INTERNAL2V56: 2.56V for Mega Boards
  - EXTERNAL: External volt supplied to AREF Pin (Pin21 internal). Limited by 5V or 3.3V depending on board type

Reading Analog Input to Computer:

```
#define AINPUT 0

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  int val = analogRead(AINPUT);
  Serial.println(val);
}
```



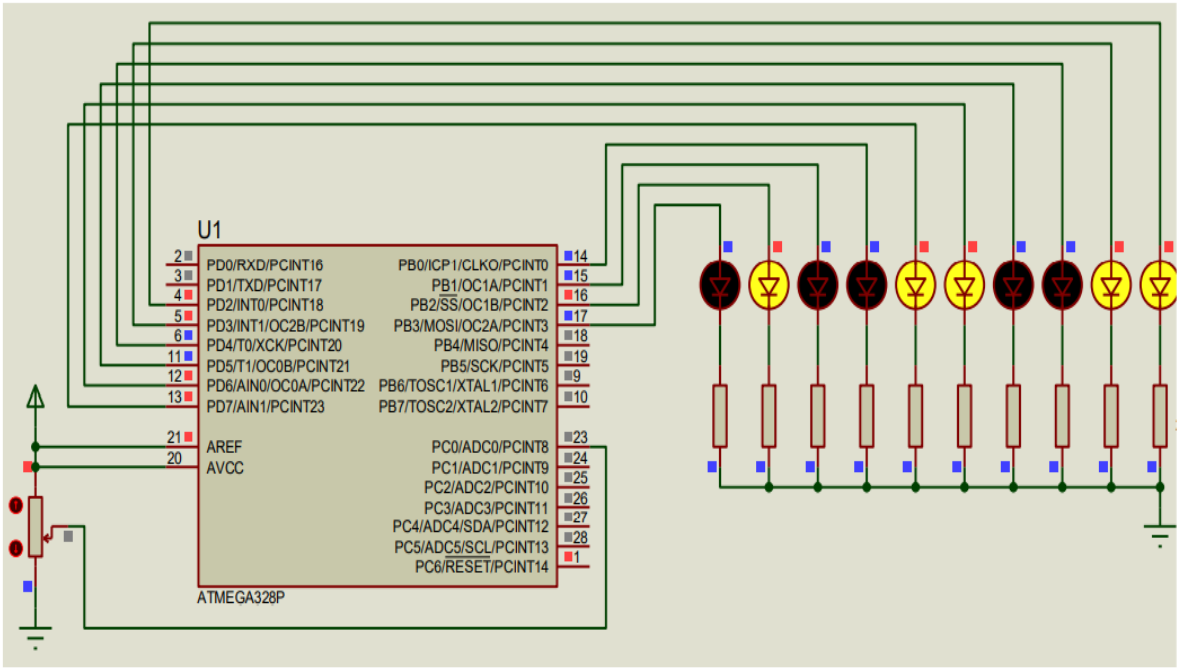
The Previous problem takes voltage divider value by changing Potentiometer value and convert it to binary number from 1 – 1023 bit

```
int val = analogRead(AINPUT);
```

analogRead(): Reads an analog number

Read Analog Input and Display its Binary Value on 10 LEDs:

```
#define AINPUT 0
#define OP 2

void setup()
{
      for(int i=0;i<10;i++)
            pinMode(OP+i, OUTPUT);
      analogReference(EXTERNAL); we put external analog reference volt
}

int value = 0;
void loop()
{
      value = analogRead(AINPUT);
      for(int i=0;i<10;i++)
      {
            digitalWrite(OP+i, value&0x1);
            value>>=1;
      }  doing & with 1 , if the bit is 1 the output will be HIGH , then we shift the number t
}          left to and make & the last bit with 1 and so on
```



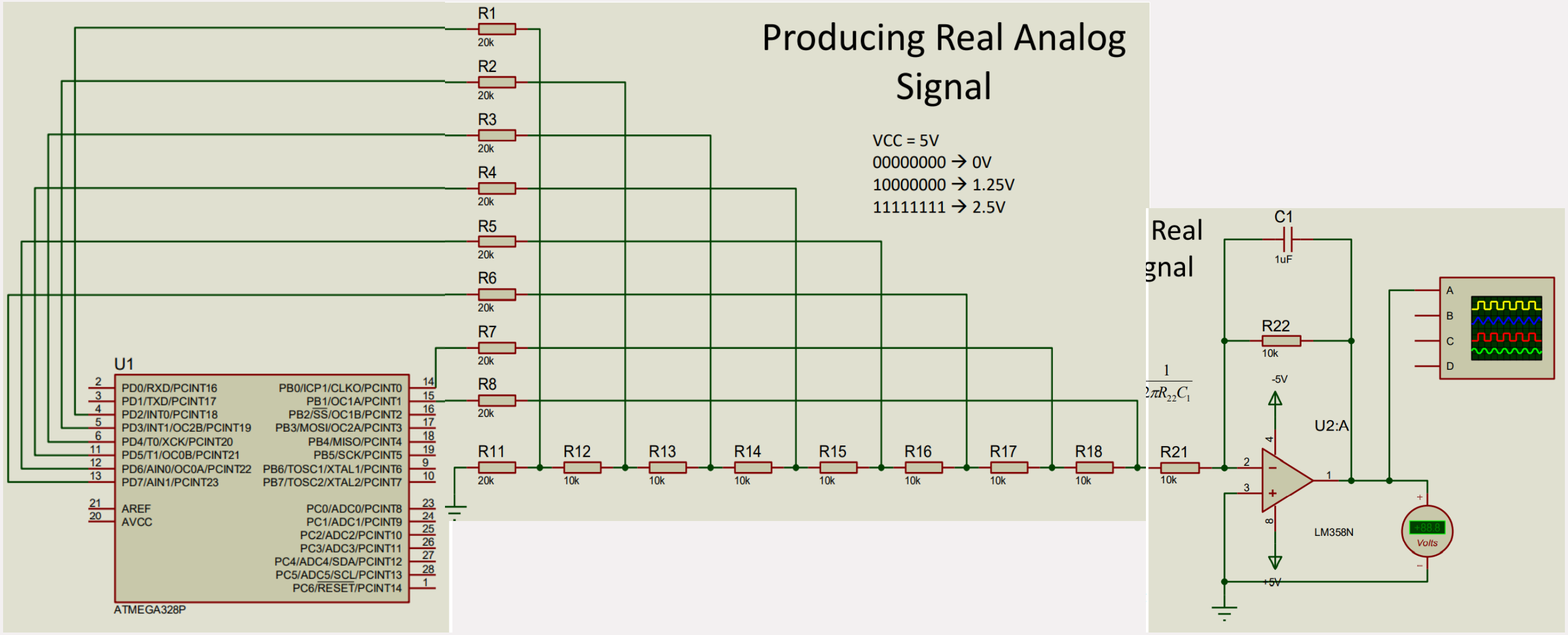
Shifting Process:

```
0000000001 <<= 1 ===> 0011011100
0011011100 >>= 1 ===> 0001101110
0001101110 >>= 1 ===> 0000110111
0000110111 >>= 1 ===> 0000011011

0000011011 &
0000000001
==============
0000000001
```

Producing Analog Signal:

- **PWM Analog-Like Signal**
- **Normal Analog Signal**

Normal Analog Signal

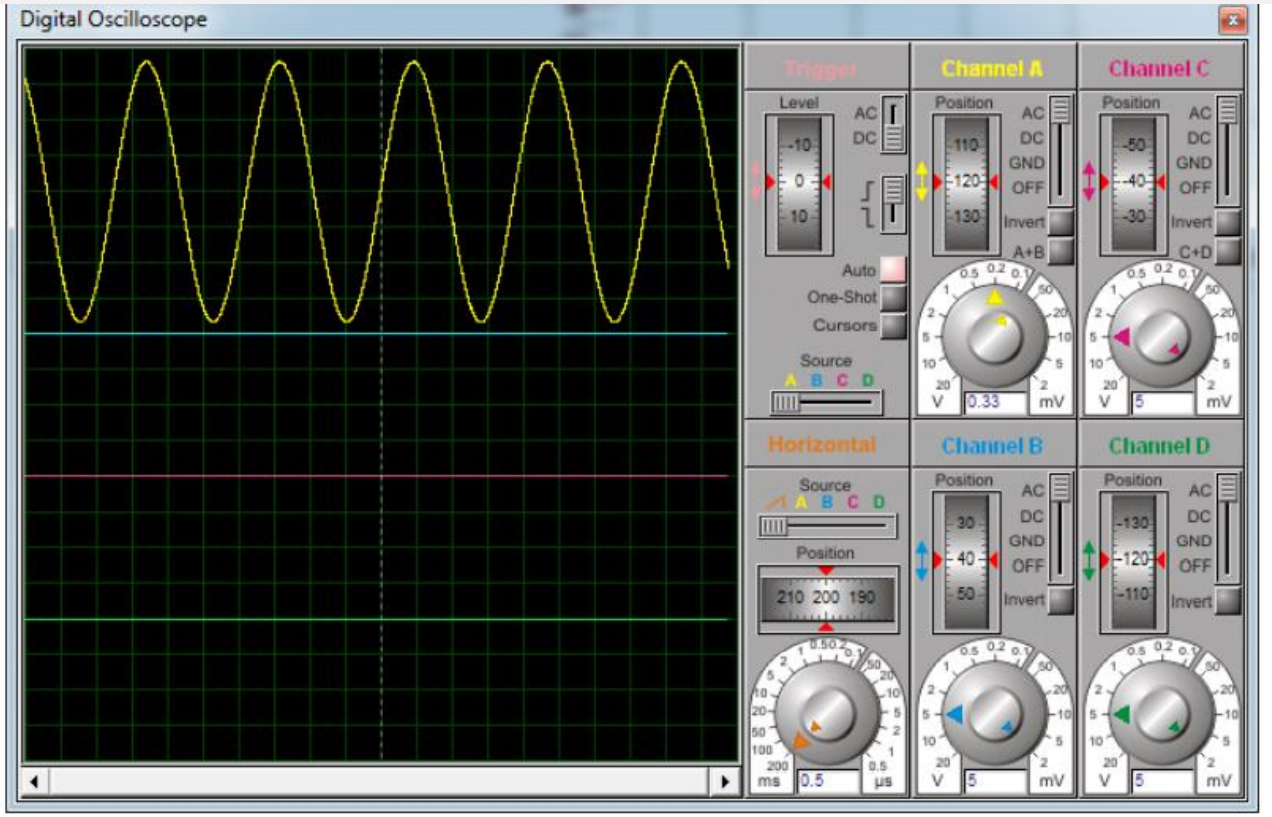


```
#define OP 2

void setup()
{
      for(int i=0;i<8;i++)
            pinMode(OP+i, OUTPUT);
}

float time = 0;
void loop()
{
      int value = 128 + 127 * sin(time);
      for(int i=0;i<8;i++)
      {
            digitalWrite(OP+i, value&0x1);
            value>>=1;
      }
      time += 0.01;
}
```

at sin(time) = 1 , value = 255
at sin(time) = 0 , value = 128
at sin(time) = -1 , value = 1

$$Gain = -\frac{R_{22}}{R_{21}}$$

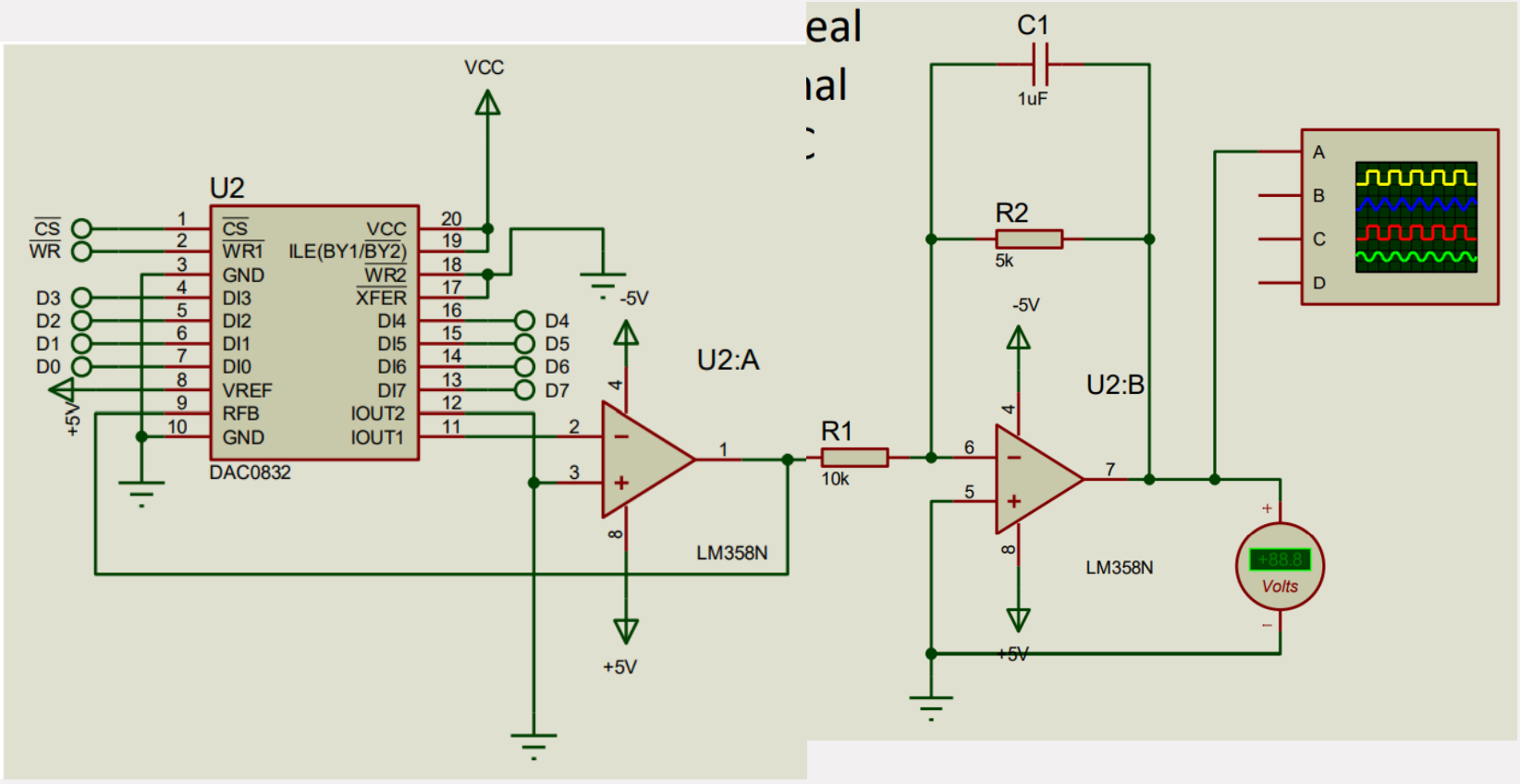
$$Frequencey_{max} = \frac{1}{2\pi R_{22} C_1}$$



Producing Real Analog Signal using DAC:

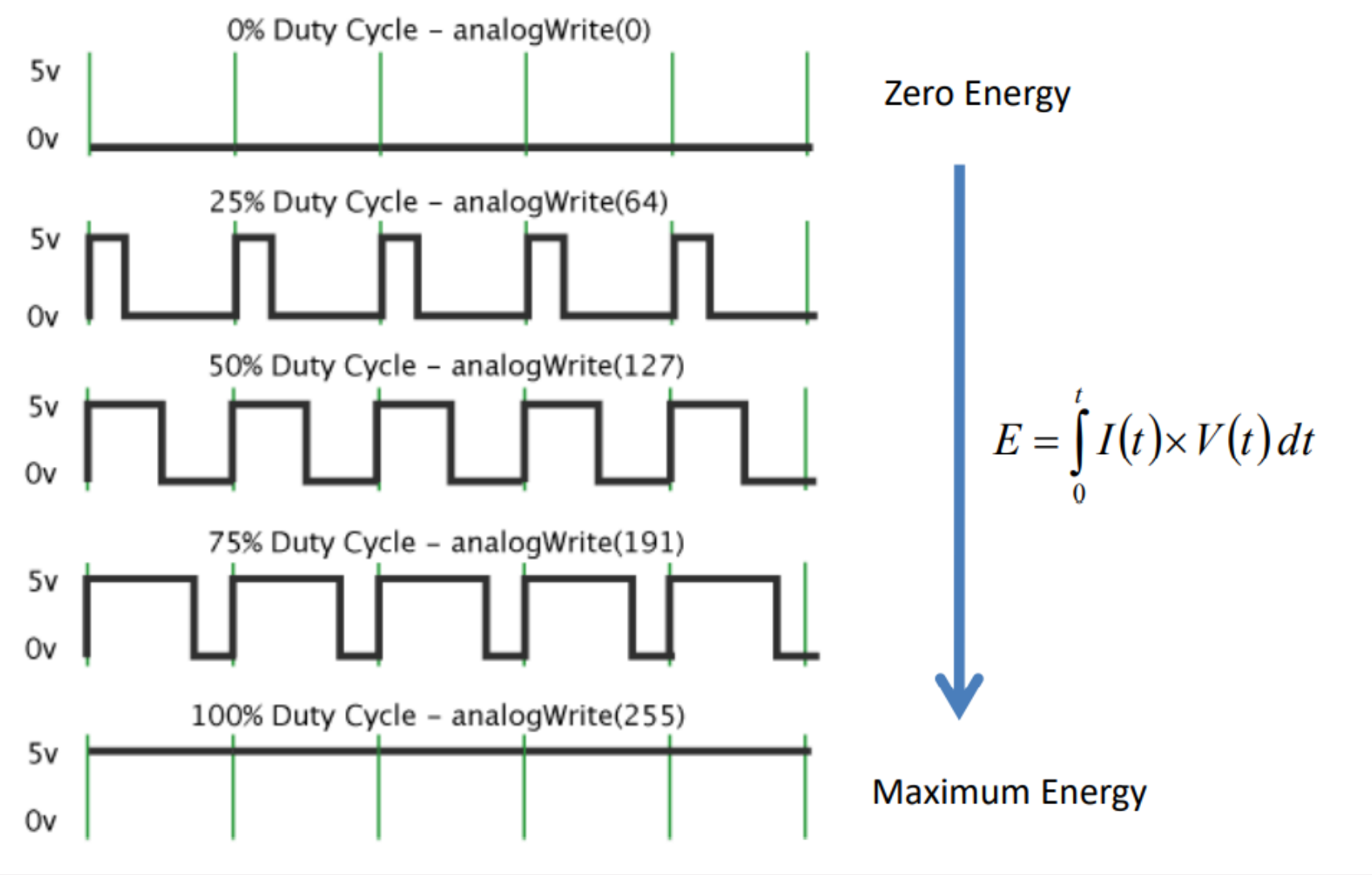
this program is same as the last program but with using a DAC chip
the advantage here that the chip has an internal buffer that took the input when I order only …
, in the first example we booked the 8 pins ALL time to produce analog signals Only.

CS: Chip Select must be 1
WR: Write State must be 1 while writing
VREF: Equals the maximum output



PWM Analog-Like Signal:

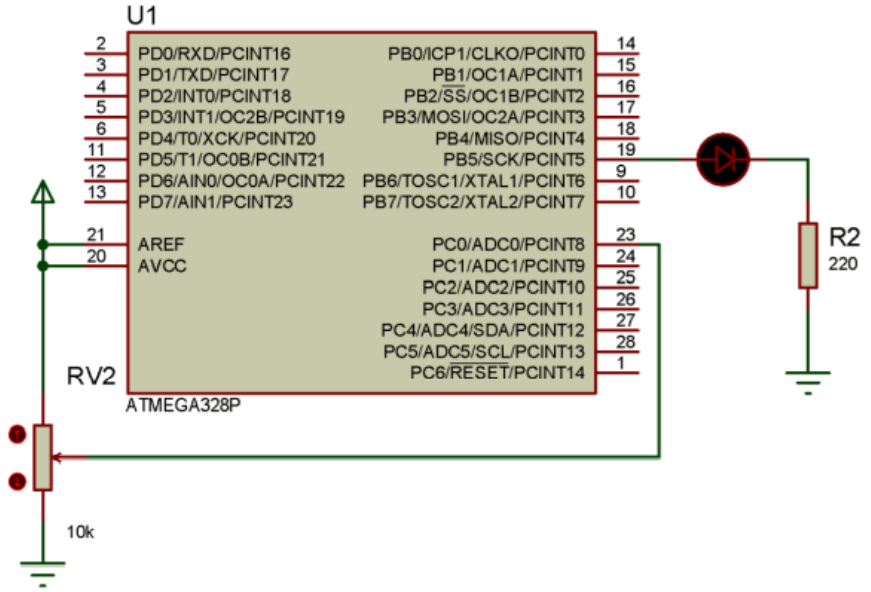
it has a big advantage that needs only 1 pin on the Arduino board not 8 bits to Run



Manual PWM:

```
#define AINPUT 0
#define LED 13

void setup()
{
      pinMode(LED, OUTPUT);
      analogReference(EXTERNAL);
}

void loop()
{
      int value = analogRead(AINPUT);
      int onTime = map(value, 0, 1023, 0, 100);

      digitalWrite(LED, HIGH);
      delay(onTime);
      digitalWrite(LED, LOW);
      delay(100-onTime);
}
```

map function : it maps the value from 1023 to 100 in this case
(value * (100/1023))



Using Built-in PWM:

Using Built-in PWM

in this program we use analogWrite on the digital pins which support PWM (pin 3,pin 5,pin 6,pin 9,pin 10,pin 11)

analogWrite(PWMPIN1 ,0); = the duty cycle is 0
analogWrite(PWMPIN1 ,64); = the duty cycle is 25%
analogWrite(PWMPIN1 ,128); = the duty cycle is 50%
analogWrite(PWMPIN1 ,192); = the duty cycle is 75%
analogWrite(PWMPIN1 ,255); = the duty cycle is 100%

Note: this method is better than the previous because there is no delay in the code as the hardware made the duty cycle by it self.

```
#define PWMPIN1 3
#define PWMPIN2 5
#define PWMPIN3 6
#define PWMPIN4 9
#define PWMPIN5 10

void setup()
{
      pinMode(PWMPIN1, OUTPUT);
      pinMode(PWMPIN2, OUTPUT);
      pinMode(PWMPIN3, OUTPUT);
      pinMode(PWMPIN4, OUTPUT);
      pinMode(PWMPIN5, OUTPUT);
}

void loop()
{
      analogWrite(PWMPIN1, 0);
      analogWrite(PWMPIN2, 64);
      analogWrite(PWMPIN3, 128);
      analogWrite(PWMPIN4, 192);
      analogWrite(PWMPIN5, 255);
}
```