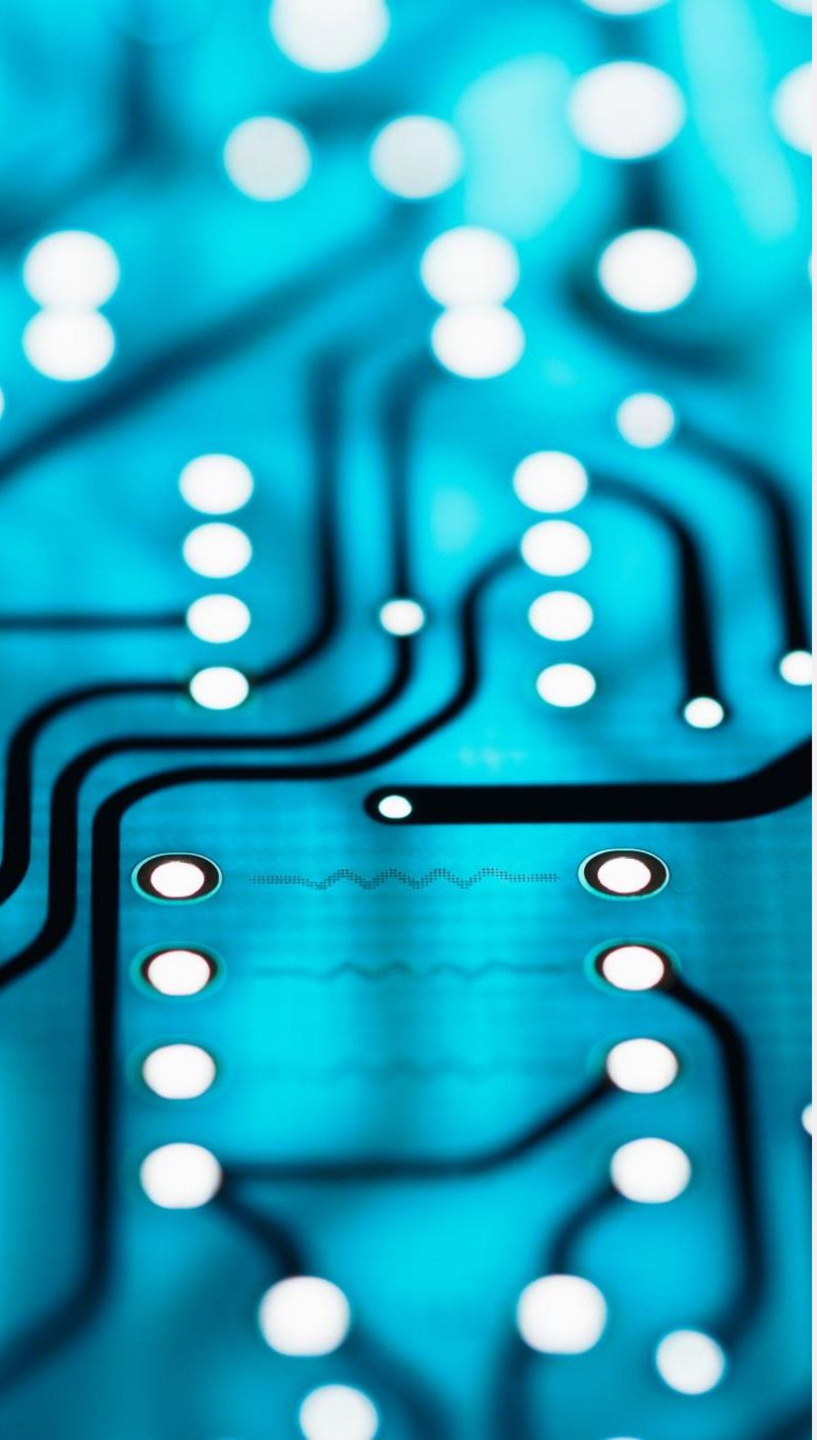


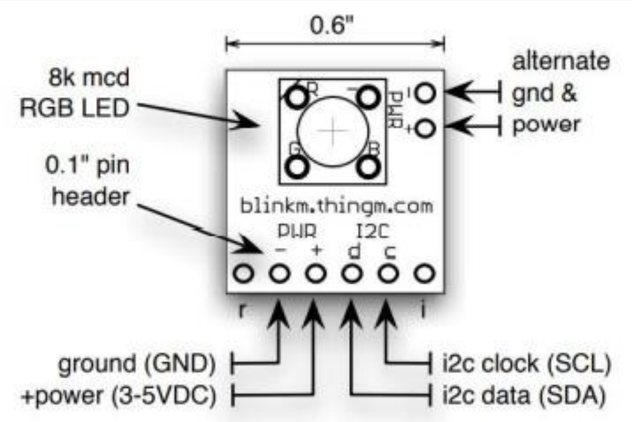
Embedded Systems (EPM)

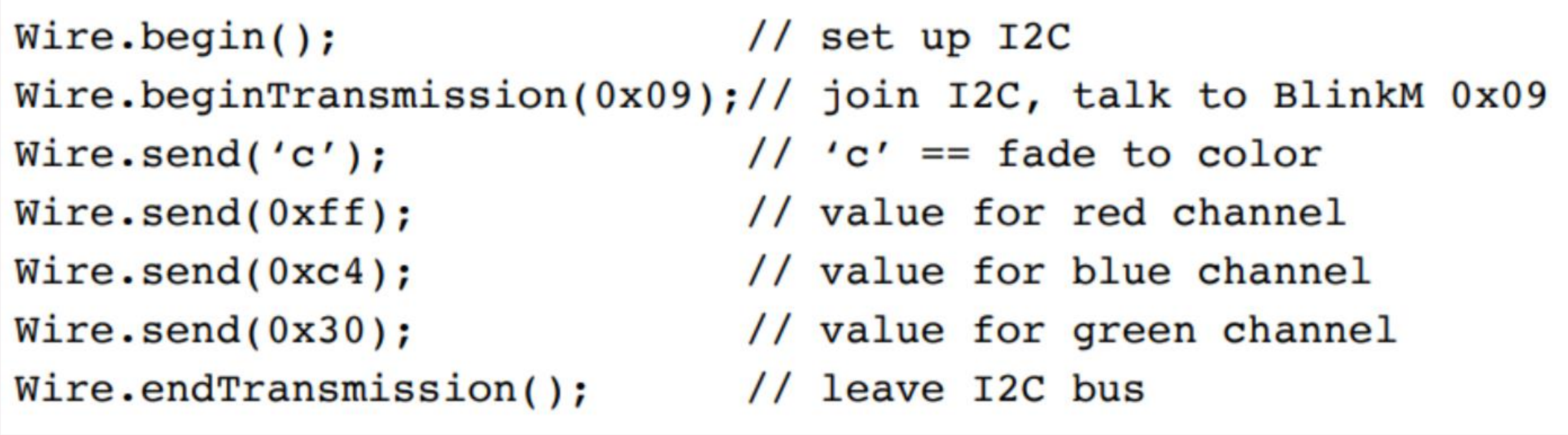
Lecture (9) Summary

1-Controlling an RGB LED Using the BlinkM Module:

Control RGB LED, Uses I2C for communication.

Configurable I2C address with default (0x00)



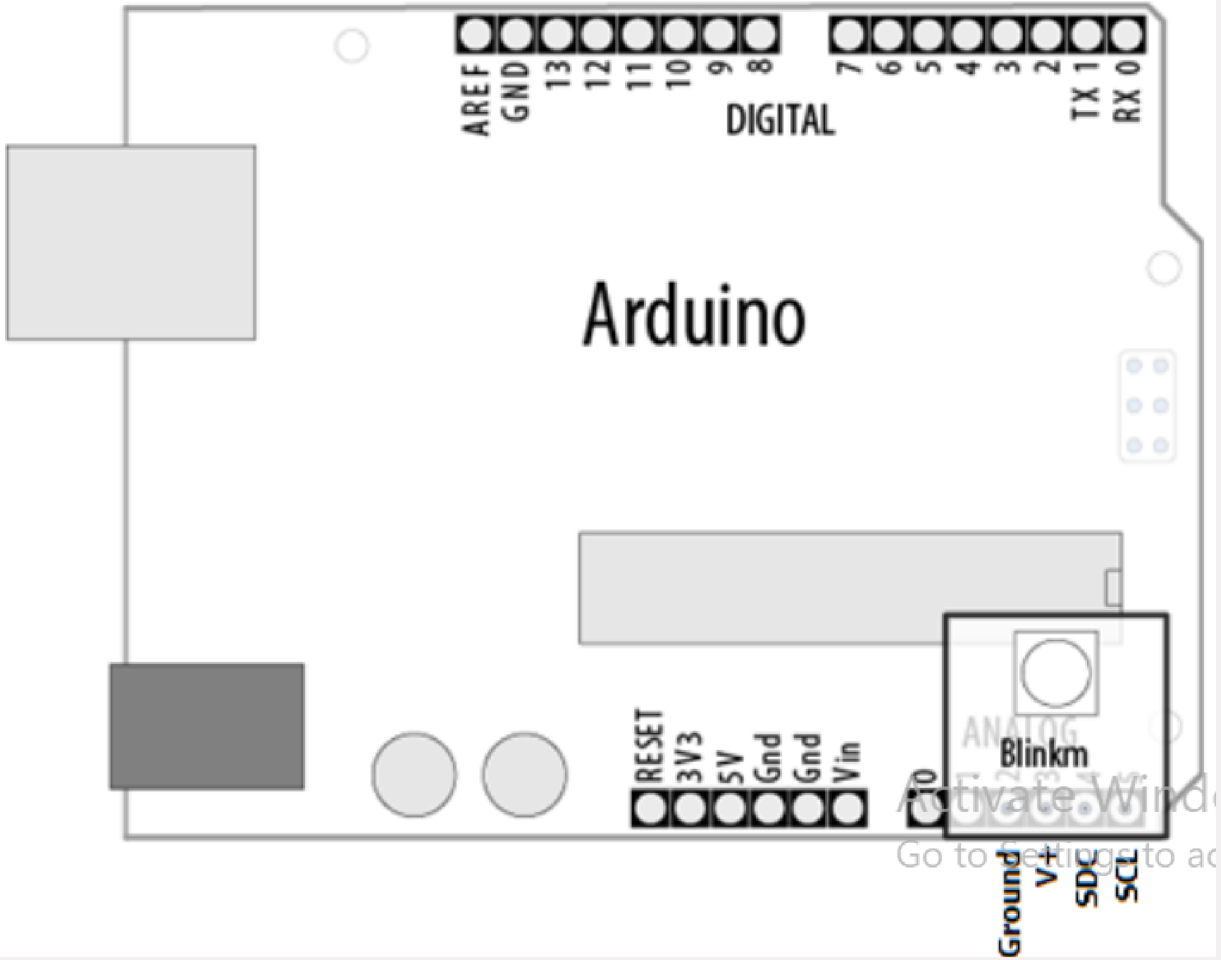


```
Wire.begin();                   // set up I2C
Wire.beginTransmission(0x09);// join I2C, talk to BlinkM 0x09
Wire.send(‘c’);                 // ‘c’ == fade to color
Wire.send(0xff);                // value for red channel
Wire.send(0xc4);                // value for blue channel
Wire.send(0x30);                // value for green channel
Wire.endTransmission();         // leave I2C bus
```

Controlling an RGB LED Using the BlinkM Module : Example

```
#include <Wire.h>
const int address = 0x00;
//I2C Address for BlinkM
byte R = 0, G = 0, B = 0;
void setup()
{
     Wire.begin();
     pinMode(16, OUTPUT);//16 Analog 2
     digitalWrite(16, LOW);//Ground
     pinMode(17, OUTPUT);//17 Analog 3
     digitalWrite(17, HIGH);//V+
}
void loop()
{
     Wire.beginTransmission(address);
     Wire.send('c');
     // 'c' == fade to color
     Wire.send(R);
     Wire.send(B);
     Wire.send(G);
     Wire.endTransmission();
     R = (R<255)?R++:255;
     if(R==255)G = (G<255)?G++:255;
     if(G==255)B = (B<255)?B++:255;
     delay(10);
}
```

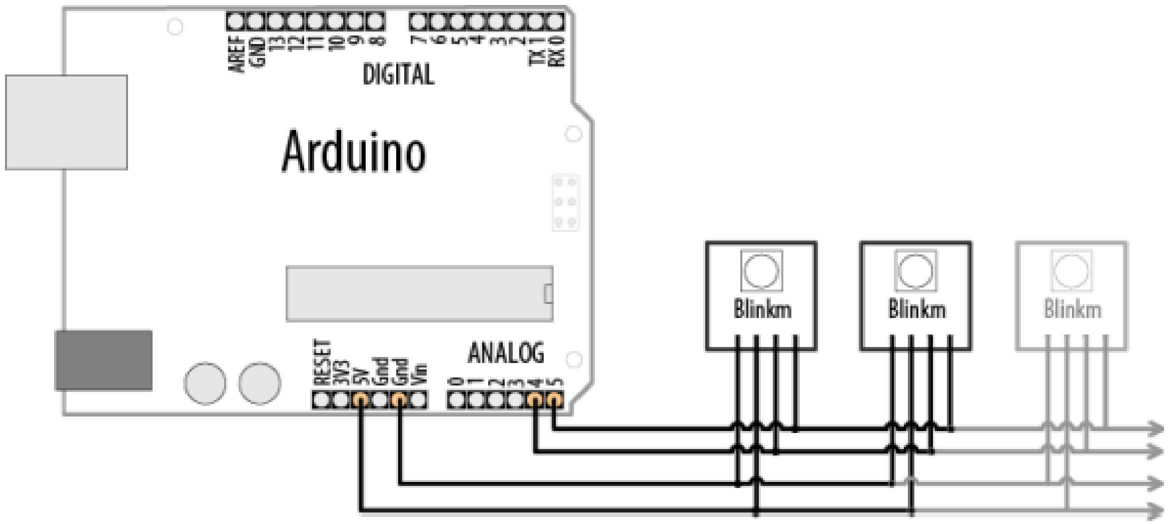
Wire.write(data)



Controlling Several BlinkM of One Address

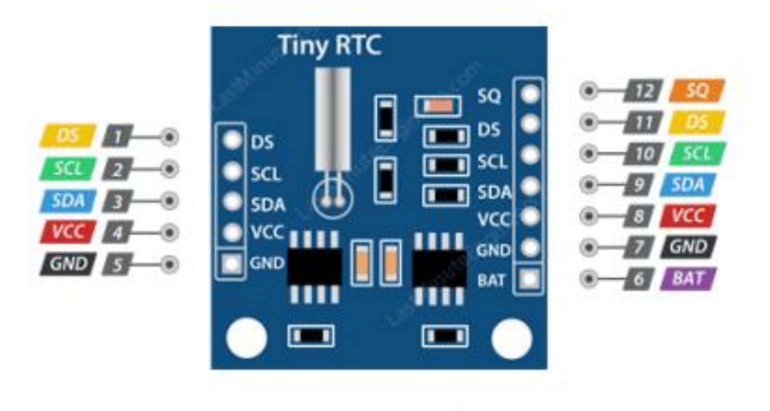
```
#include <Wire.h>
int addressA = 9;
int addressB = 10;
int addressC = 11;
byte R = 125, G = 64, B = 225;
void setup()
{
     Wire.begin();
}
void setColor(int address, byte R, byte G, byte B)
{
     Wire.beginTransmission(address);
     Wire.send('c');
     Wire.send(R);
     Wire.send(B);
     Wire.send(G);
     Wire.endTransmission();
}
void loop()
{
     setColor(addressA, R, G, B);
     setColor(addressB, G, B, R);
     setColor(addressA, B, R, G);
     delay(10);
}
```

Using configuration kit for BlinkM module you can set the device I2C address from computer using serial interface.



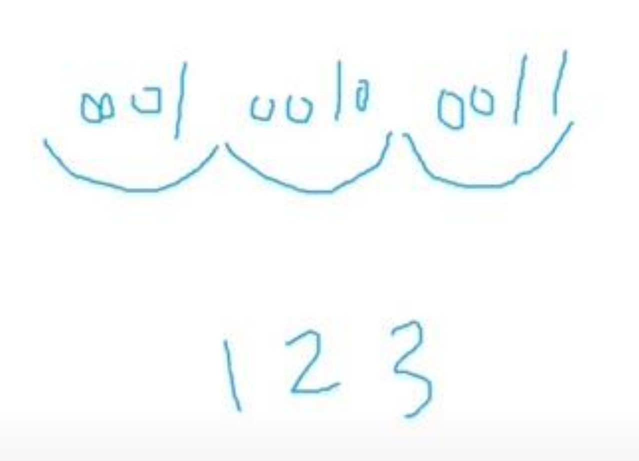
2-Using External Real Time Clock Module:

Produce Real Time Clock
Uses I2C for communication.



Produce 7 BCD values for (second, minute, hour, week, day, month, year -2000)
Example of BCD value (0x25)16 → (25)10

Example on Binary Coded Decimal:



Configurable I2C address with default (0x68)

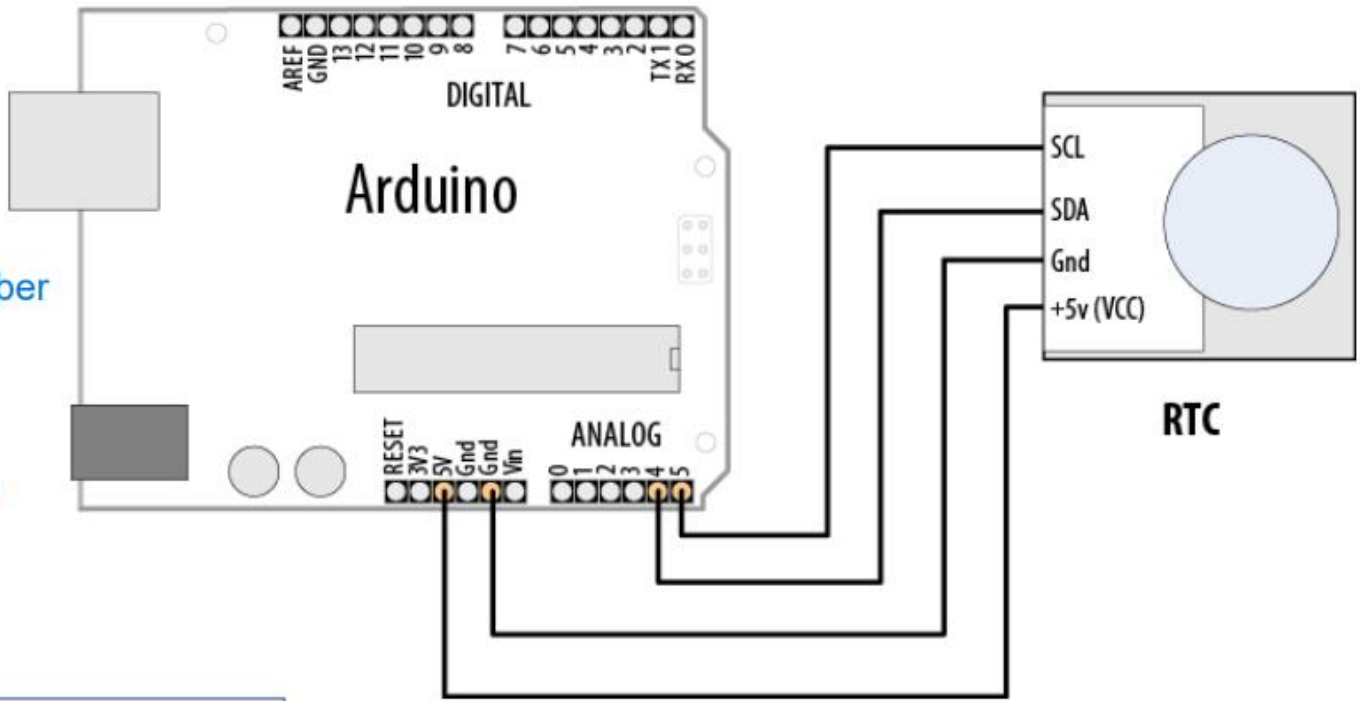
Using External Real Time Clock Module Example

```
#include <Wire.h>
const byte RTCAddress = 0x68;
int second, minute, hour, day, wDay, month, year;
void setup() {
      Serial.begin(9600);
      Wire.begin();
}
byte bcd2dec(byte n){return (n/16)*10 + (n%16);}
void loop() {
      //Initialize RTC by sending 0
      Wire.beginTransmission(RTCAddress);
      Wire.send(0);
      Wire.endTransmission();
      //Request 7 fields (each 1 byte)
      Wire.requestFrom(RTCAddress, (byte)7);
      second = bcd2dec(Wire.receive() & 0x7f);
      minute = bcd2dec(Wire.receive());
      hour = bcd2dec(Wire.receive()&0x3f);
      wDay = bcd2dec(Wire.receive());
      day = bcd2dec(Wire.receive());
      month = bcd2dec(Wire.receive());
      year = bcd2dec(Wire.receive()) + 2000;
      String s;
      s = s + day + "/" + month + "/" + year + " ";
      s = s + hour + ":" + minute + ":" + second;
      Serial.println(s);
      delay(1000);
}
```

function takes bcd number and return it as binary

want to read 7 things

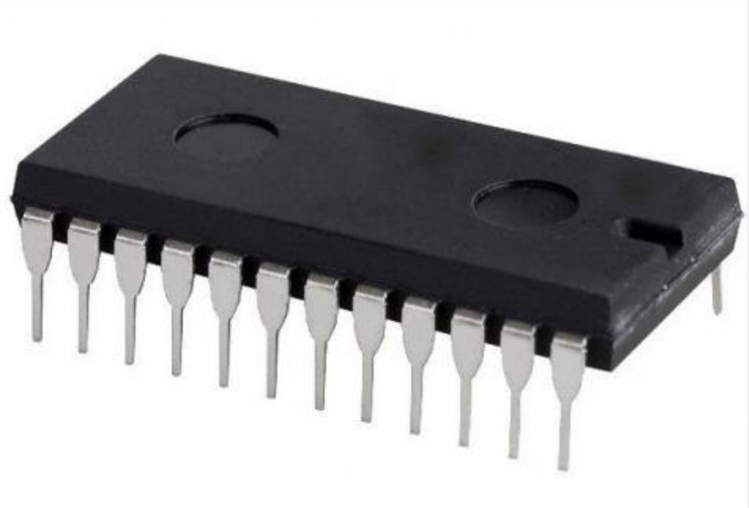
Wire.read()



3-Driving Four 7-Segment LEDs Using Only Two Wires:

LED Driver Module
Uses I2C for communication.
Default Address: 0x38

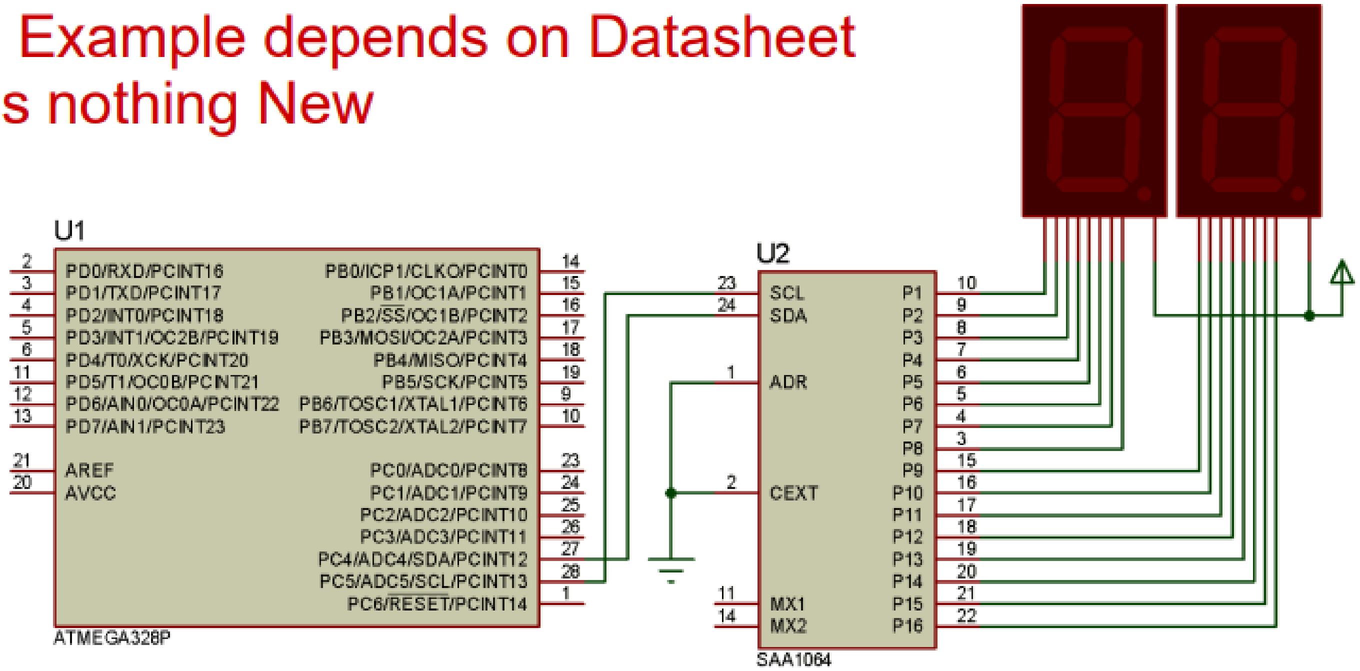
note: if using SPI then we use more than 2 wires



Driving Four 7-Segment LEDs Using Only Two Wires

```
#include "Wire.h"  // enable I2C bus
byte address = 0x38;
int digits[16]={63, 6, 91, 79, 102, 109, 125,7,
               127, 111, 119, 124, 57, 94, 121, 113};
void setup() {
     Wire.begin(); // start up I2C bus
     delay(100);
     Wire.beginTransmission(address);
     Wire.send(B00000000);
     //Zero means the next byte is the control byte
     Wire.send(B01000000);
     //Control Byte: static mode on, 12mA segment current
     Wire.endTransmission();
}
void loop() {
     static int i = 0;
     Wire.beginTransmission(address);
     Wire.send(1);
     //1 means data mode
     Wire.send(digits[(i+0)%16]); // digit 1 (RHS)
     Wire.send(digits[(i+1)%16]); // digit 2
     Wire.endTransmission();
     delay(100);
     i++;
}
```

This Example depends on Datasheet
it has nothing New



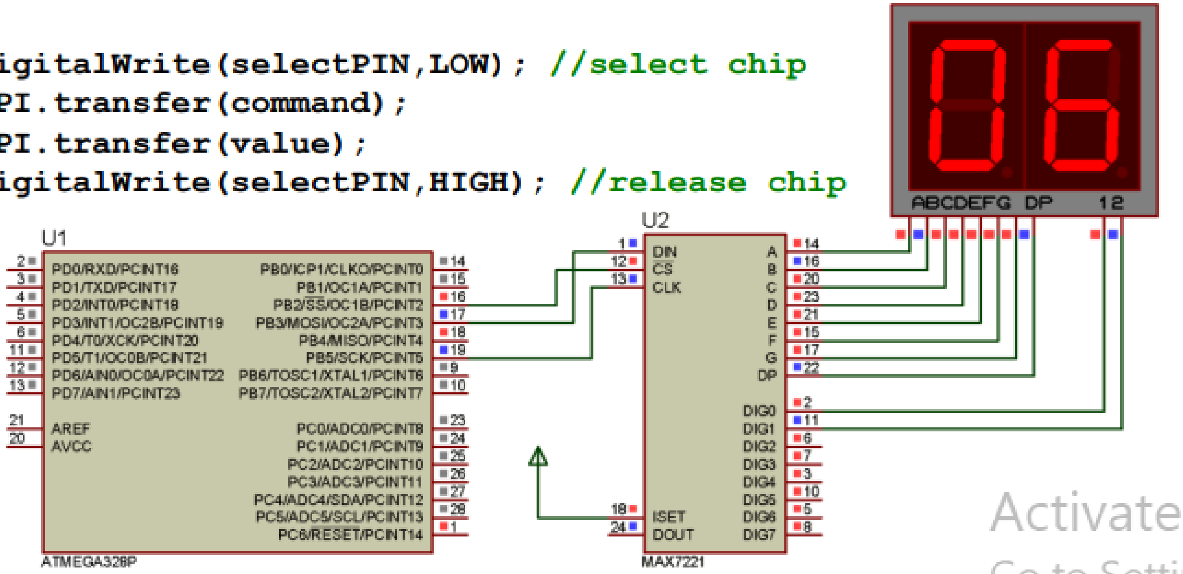
Driving Multidigit, 7-Segment Displays Using SPI

```
#include <SPI.h>
const int selectPIN = 10;
const int nDigits = 2;
const int maxValue = 99;
void setup()
{
      SPI.begin(); // initialize SPI
      pinMode(selectPIN, OUTPUT);
      digitalWrite(selectPIN,LOW); //select slave
      sendCommand(12,1); // normal mode
      sendCommand(15,0); // display test off
      sendCommand(10,8); // set medium intensity
      sendCommand(11, nDigits); // 2 digits
      sendCommand(9,255); // standard 7 Segment digits
      digitalWrite(selectPIN,HIGH); //deselect slave
}
void loop()
{
      static int i = 0;
      displayNumber(i, nDigits);
      i = (i>maxValue)?0:(i+1);
      delay(25);
}
```

from Data Sheet

the counter will count from 0 -> 99 until i becomes 100 the counter will be 00 again

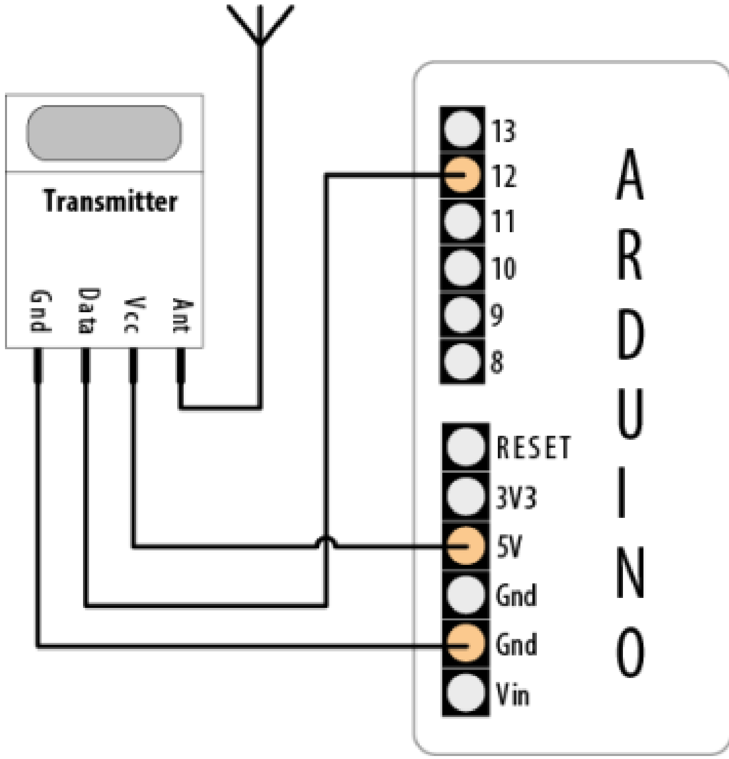
```
void displayNumber(int number, int nDigits)
{
      for(int i = 0;i<nDigits;i++)
      {
            byte character = number % 10;
            sendCommand(nDigits-i, character);
            number = number / 10;
      }
}
void sendCommand(int command, int value)
{
      digitalWrite(selectPIN,LOW); //select chip
      SPI.transfer(command);
      SPI.transfer(value);
      digitalWrite(selectPIN,HIGH); //release chip
}
```



RF Communication (RF Transmitter)

4-Radio Frequency Communication

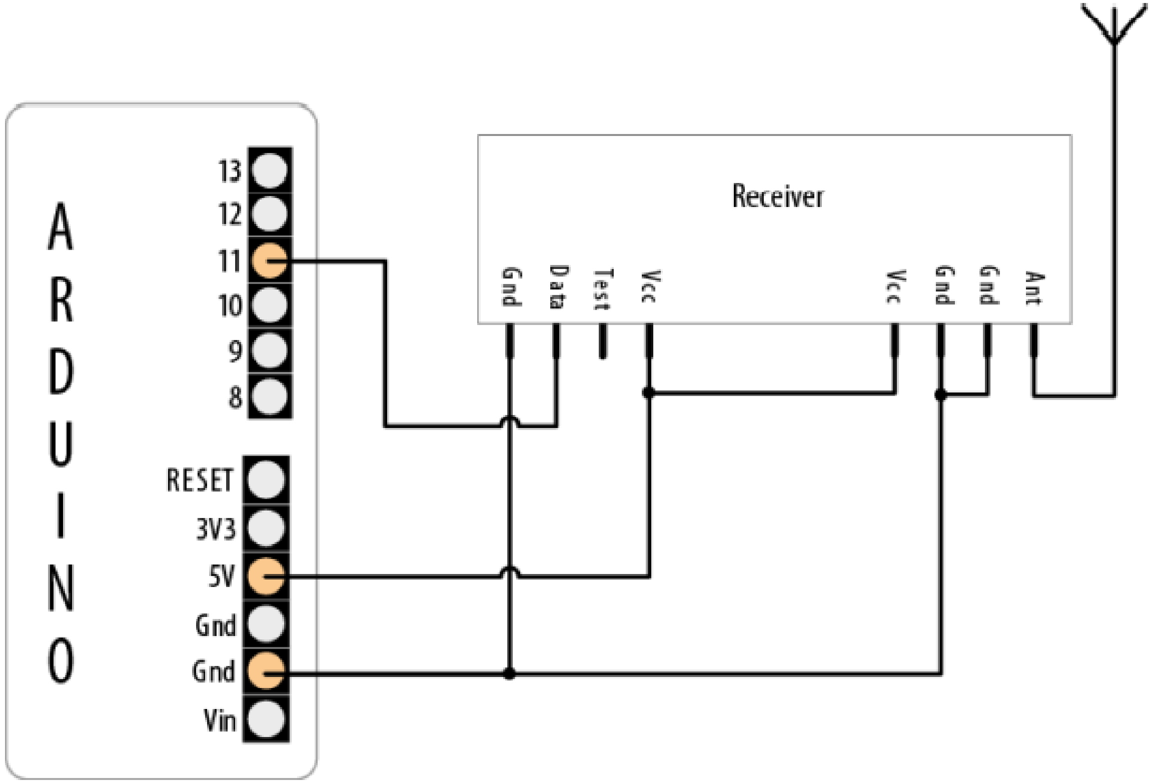
```
#include <VirtualWire.h>
void setup()
{
     // Initialize the IO and ISR
     vw_setup(2000); // Bits per sec
}
void loop()
{
     send("hello");
     delay(1000);
}
void send (char *message)
{              casting to unsigned int   Length of message
     vw_send((uint8_t *)message, strlen(message));
     vw_wait_tx(); // Wait until the whole message is gone
}
```



RF Communication (RF Receiver)

```
#include <VirtualWire.h>
byte message[VW_MAX_MESSAGE_LEN];
byte msgLength = VW_MAX_MESSAGE_LEN;
void setup(){
     Serial.begin(9600);
     Serial.println("Ready");
     vw_setup(2000); same rate choosed in transmitting
     vw_rx_start();
}
void loop(){  waiting until receiving message
     if (vw_get_message(message, &msgLength)){
          Serial.print("Got: ");
          for (int i = 0; i < msgLength; i++)
               Serial.write(message[i]);
          Serial.println();
     }
}
```

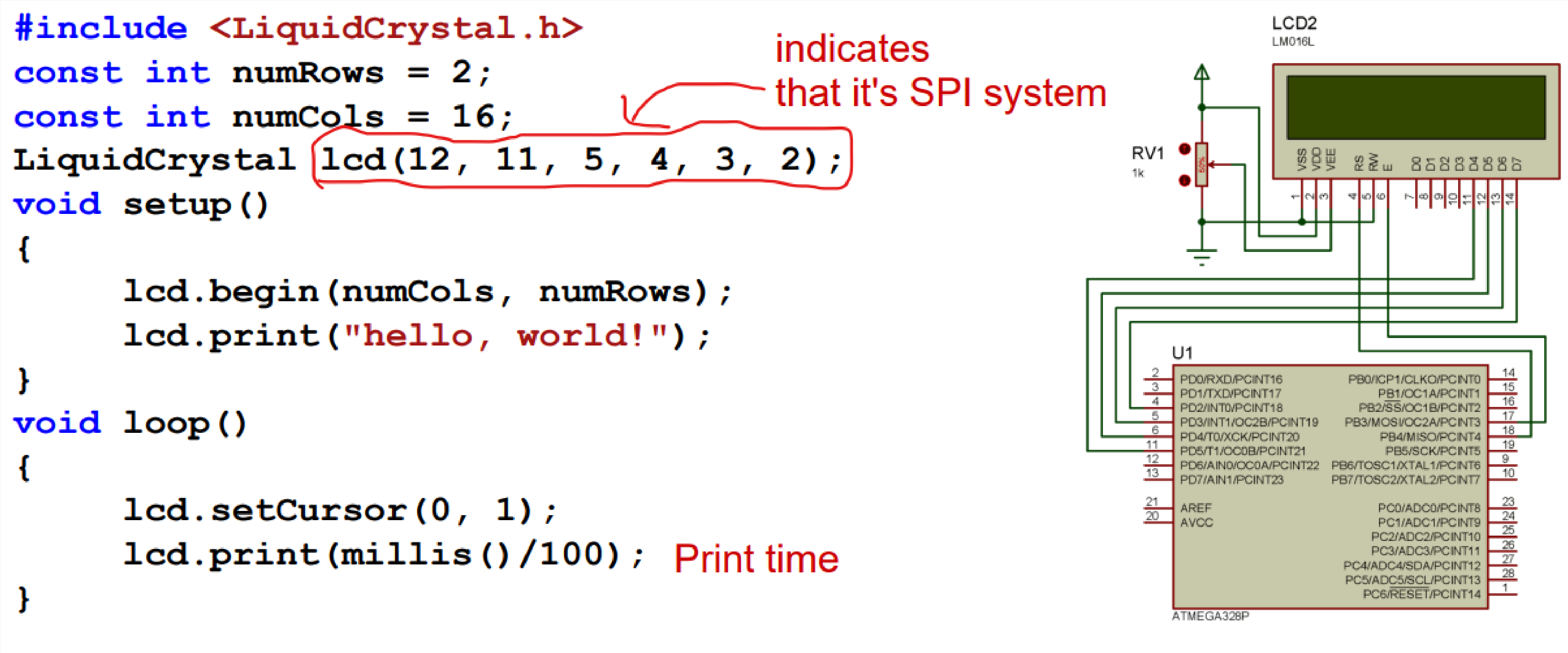
Note: If we need a fast rate device we choose SPI
If we need a slow rate device we choose I2C



5-Display Devices:

(a)LCD Text Display:

LCD: Liquid Crystal Display, Uses industry standard HD44780 , Uses serial communication



```
#include <LiquidCrystal.h>
const int numRows = 2;
const int numCols = 16;
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
void setup()
{
      lcd.begin(numCols, numRows);
      lcd.print("hello, world!");
}
void loop()
{
      lcd.setCursor(0, 1);
      lcd.print(millis()/100);
}
```

indicates
that it's SPI system

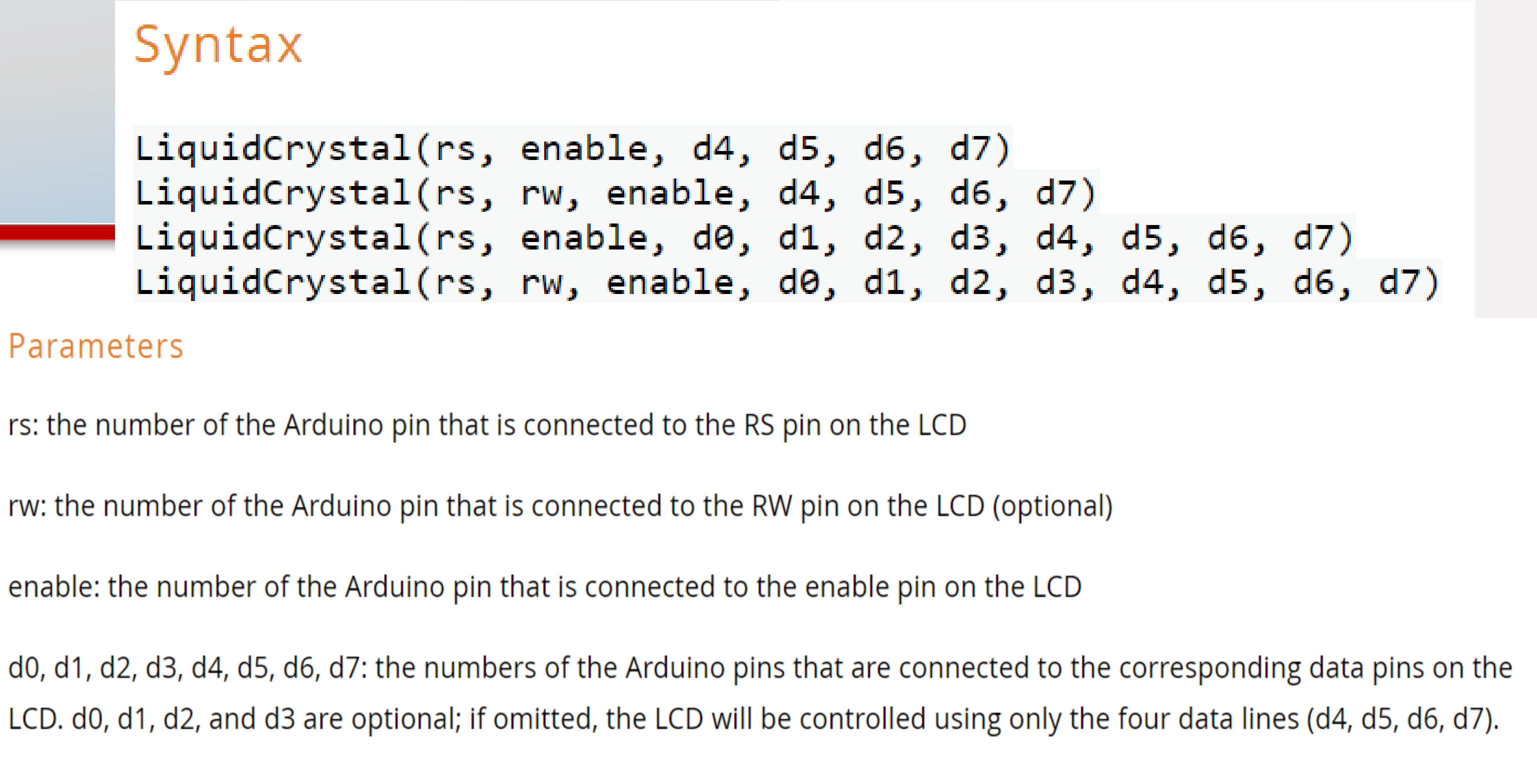
Print time

LCD Text Display: Scrolling Text

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
const int numRows = 2;
const int numCols = 16;
const char textString[] = "Hello World";
const int textLength = sizeof(textString) -1;
void setup()
{
     lcd.begin(numCols, numRows);
     lcd.print(textString);
}
void loop()
{
     for(int i=0;i<textLength;i++)
     {
          lcd.scrollDisplayRight();
          delay(20);
     }
     for(int i=0;i<textLength;i++)
     {
          lcd.scrollDisplayLeft();
          delay(20);
     }
}
```

Scroll the text to Right Every 20 ms

Scroll the text to Left Every 20 ms



Syntax

```
LiquidCrystal(rs, enable, d4, d5, d6, d7)
LiquidCrystal(rs, rw, enable, d4, d5, d6, d7)
LiquidCrystal(rs, enable, d0, d1, d2, d3, d4, d5, d6, d7)
LiquidCrystal(rs, rw, enable, d0, d1, d2, d3, d4, d5, d6, d7)
```

Parameters

rs: the number of the Arduino pin that is connected to the RS pin on the LCD

rw: the number of the Arduino pin that is connected to the RW pin on the LCD (optional)

enable: the number of the Arduino pin that is connected to the enable pin on the LCD

d0, d1, d2, d3, d4, d5, d6, d7: the numbers of the Arduino pins that are connected to the corresponding data pins on the LCD. d0, d1, d2, and d3 are optional; if omitted, the LCD will be controlled using only the four data lines (d4, d5, d6, d7).

LCD Text Display: Displaying Special Symbols

```
#include <LiquidCrystal.h>
const int numRows = 2;
const int numCols = 16;
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
void setup()
{
     lcd.begin(numRows, numCols);
     showSymbol(B11011111, "degrees");
     showSymbol (B11110111, "pi");
     showSymbol(B11101100, "cents");
     showSymbol(B11101000, "sqrt");
     showSymbol(B11110100, "ohms");
     lcd.clear();
}
void loop(){}
void showSymbol( byte symbol, char * description)
{
     lcd.clear();
     lcd.print(symbol);
     lcd.print(' ');
     lcd.print(description);
     delay(200);
}
```

This function took the symbol (which stored in the library) and description and print it on the LCD

LCD Text Display:
Creating Custom Characters

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
byte happy[8]={     B00000, B10001, B00000, B00000,
B10001,   B01110,   B00000, B00000 };
byte saddy[8]={ B00000, B10001, B00000, B00000,
B01110, B10001, B00000, B00000      };
void setup() {
     lcd.createChar(0, happy);
     lcd.createChar(1, saddy);
     lcd.begin(16, 2);
}
void loop() {
     for (int i=0; i<2; i++)
     {
          lcd.setCursor(0,0);
          lcd.write(i);
          lcd.print(" hello");
          delay(500);
     }
}
```

Creating a character enterd by user

happy face:

0	0	0	0	0
1	0	0	0	1
0	0	0	0	0
0	0	0	0	0
1	0	0	0	1
0	1	1	1	0
0	0	0	0	0
0		0	0	0

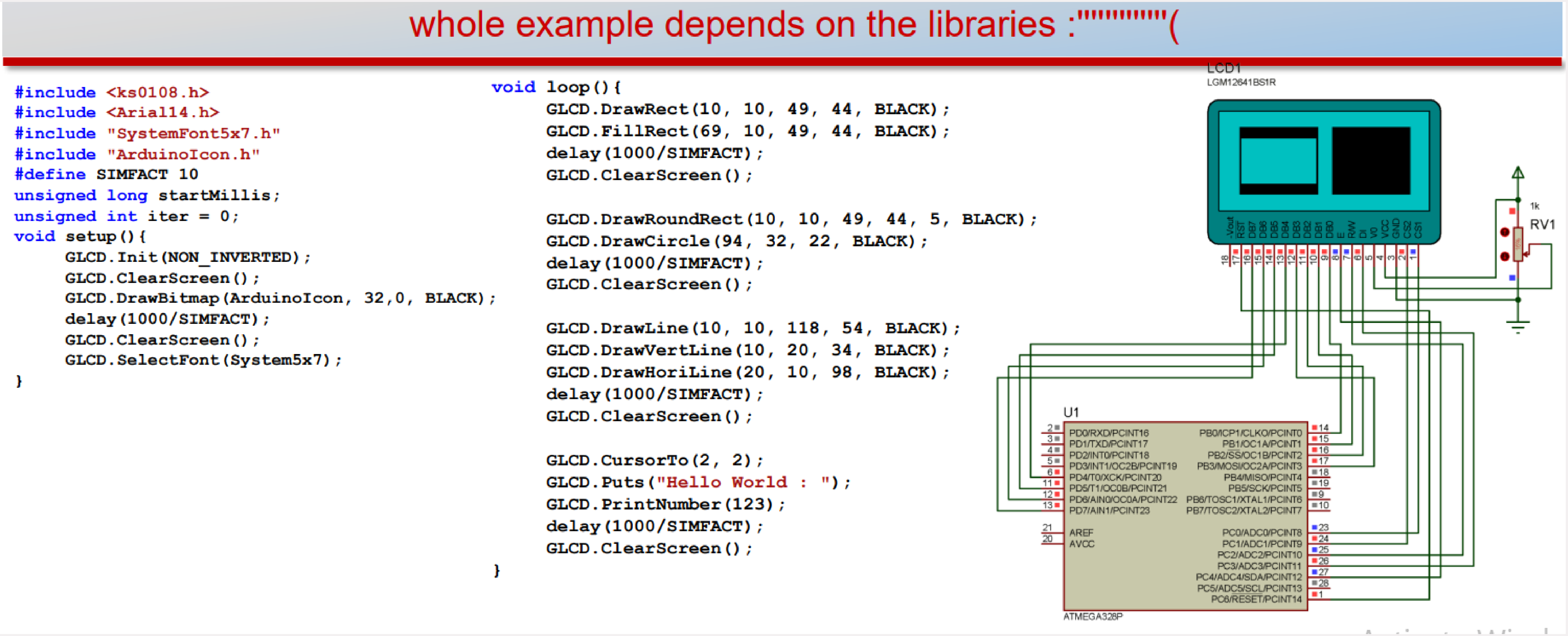
Sad face:

0	0	0	0	0
1	0	0	0	1
0	0	0	0	0
0	0	0	0	0
0	1	1	1	0
1	0	0	0	1
0	0	0	0	0
0	0	0	0	0

(b) LCD Graphics Display:

LCD Graphics: Liquid Crystal Display with Graphics Support

Uses industry standard KS0108 ,Uses serial communication



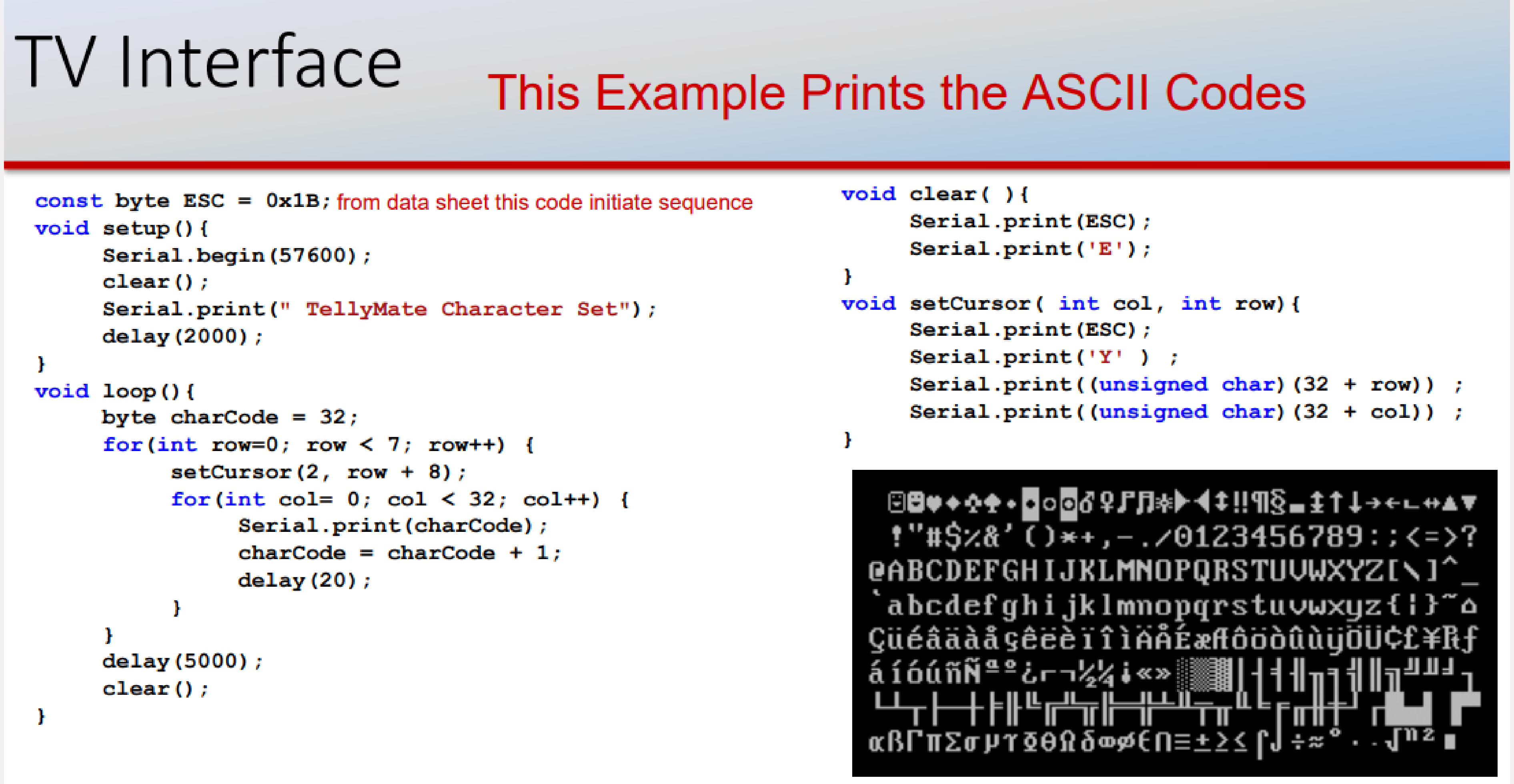
whole example depends on the libraries :'''''''''''''(

```
#include <ks0108.h>
#include <Arial14.h>
#include "SystemFont5x7.h"
#include "ArduinoIcon.h"
#define SIMFACT 10
unsigned long startMillis;
unsigned int iter = 0;
void setup(){
     GLCD.Init(NON_INVERTED);
     GLCD.ClearScreen();
     GLCD.DrawBitmap(ArduinoIcon, 32,0, BLACK);
     delay(1000/SIMFACT);
     GLCD.ClearScreen();
     GLCD.SelectFont(System5x7);
}
```

```
void loop(){
     GLCD.DrawRect(10, 10, 49, 44, BLACK);
     GLCD.FillRect(69, 10, 49, 44, BLACK);
     delay(1000/SIMFACT);
     GLCD.ClearScreen();

     GLCD.DrawRoundRect(10, 10, 49, 44, 5, BLACK);
     GLCD.DrawCircle(94, 32, 22, BLACK);
     delay(1000/SIMFACT);
     GLCD.ClearScreen();

     GLCD.DrawLine(10, 10, 118, 54, BLACK);
     GLCD.DrawVertLine(10, 20, 34, BLACK);
     GLCD.DrawHoriLine(20, 10, 98, BLACK);
     delay(1000/SIMFACT);
     GLCD.ClearScreen();

     GLCD.CursorTo(2, 2);
     GLCD.Puts("Hello World : ");
     GLCD.PrintNumber(123);
     delay(1000/SIMFACT);
     GLCD.ClearScreen();
}
```

(c)TV interface:

Produce Analog Video Signal to TV ,Controlled by Serial Interface



TV Interface

This Example Prints the ASCII Codes

```
const byte ESC = 0x1B; from data sheet this code initiate sequence
void setup(){
     Serial.begin(57600);
     clear();
     Serial.print(" TellyMate Character Set");
     delay(2000);
}
void loop(){
     byte charCode = 32;
     for(int row=0; row < 7; row++) {
          setCursor(2, row + 8);
          for(int col= 0; col < 32; col++) {
               Serial.print(charCode);
               charCode = charCode + 1;
               delay(20);
          }
     }
     delay(5000);
     clear();
}
```

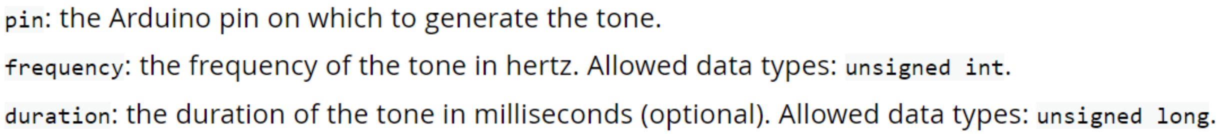
```
void clear( ){
     Serial.print(ESC);
     Serial.print('E');
}
void setCursor( int col, int row){
     Serial.print(ESC);
     Serial.print('Y' ) ;
     Serial.print((unsigned char)(32 + row)) ;
     Serial.print((unsigned char)(32 + col)) ;
}
```

```
 ☺☻♥♦♣♠•◘○◙♂♀♪♫☼►◄↕‼¶§▬↨↑↓→←∟↔▲▼
 !"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
`abcdefghijklmnopqrstuvwxyz{¦}~⌂
ÇüéâäàåçêëèïîìÄÅÉæÆôöòûùÿÖÜ¢£¥₧ƒ
áíóúñÑªº¿⌐¬½¼¡«»░▒▓│┤╡╢╖╕╣║╗╝╜╛┐
└┴┬├─┼╞╟╚╔╩╦╠═╬╧╨╤╥╙╘╒╓╫╪┘┌█▄▌▐▀
αßΓπΣσµτΦΘΩδ∞φε∩≡±≥≤⌠⌡÷≈°∙·√ⁿ²■
```

Playing Tones

```
const int speakerPin = 9;
const int pitchPin = 0;
const int durationPin = 1;
void setup(){
}
void loop(){
     int sensor0Reading = analogRead(pitchPin);
     int sensor1Reading = analogRead(durationPin);
     int frequency = map(sensor0Reading, 0, 1023, 100,5000);
     int duration = map(sensor1Reading, 0, 1023, 100,1000);
     tone(speakerPin, frequency, duration);
     delay(duration);
}
```

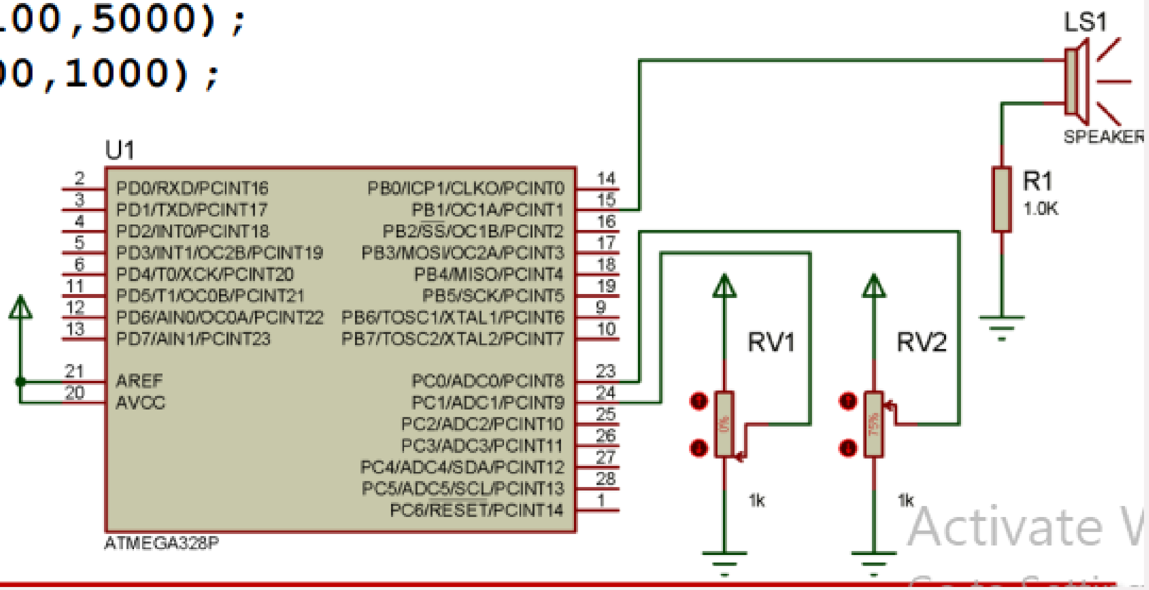
Reads the values from analog inputs



`pin`: the Arduino pin on which to generate the tone.

`frequency`: the frequency of the tone in hertz. Allowed data types: `unsigned int`.

`duration`: the duration of the tone in milliseconds (optional). Allowed data types: `unsigned long`.

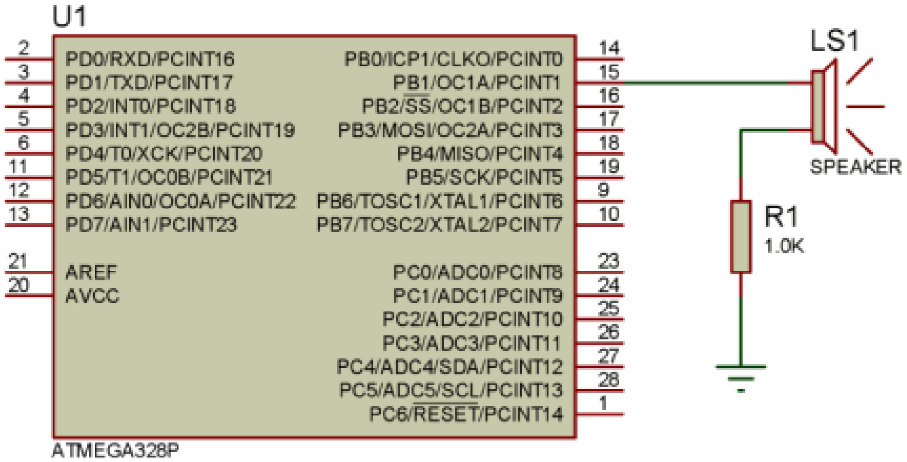


Playing a Simple Melody

Make a music

```
#define SIMFACT 10//10: Simulator 1:Real
const int speakerPin = 9;
char noteNames[] = {'C','D','E','F','G','a','b'};
unsigned int frequencies[] = {262,294,330,349,392,440,494};
const byte noteCount = sizeof(noteNames);
char score[] = "CCGGaaGFFEEDDC GGFFEEDGGFFEED CCGGaaGFFEEDDC ";
const byte scoreLen = sizeof(score);
void setup(){}
void loop(){
     for (int i = 0; i < scoreLen; i++){
          int duration = 333;
          playNote(score[i], duration);
     }
     delay(4000/SIMFACT);
}
void playNote(char note, int duration){
     for (int i = 0; i < noteCount; i++){
          if (noteNames[i] == note)
               tone(speakerPin, frequencies[i]*SIMFACT, duration/SIMFACT); }
     delay(duration/SIMFACT);
}
```

function that makes a tone on every charachter


U1
2 PD0/RXD/PCINT16
3 PD1/TXD/PCINT17
4 PD2/INT0/PCINT18
5 PD3/INT1/OC2B/PCINT19
6 PD4/T0/XCK/PCINT20
11 PD5/T1/OC0B/PCINT21
12 PD6/AIN0/OC0A/PCINT22
13 PD7/AIN1/PCINT23
21 AREF
20 AVCC
14 PB0/ICP1/CLKO/PCINT0
15 PB1/OC1A/PCINT1
16 PB2/SS/OC1B/PCINT2
17 PB3/MOSI/OC2A/PCINT3
18 PB4/MISO/PCINT4
19 PB5/SCK/PCINT5
9 PB6/TOSC1/XTAL1/PCINT6
10 PB7/TOSC2/XTAL2/PCINT7
23 PC0/ADC0/PCINT8
24 PC1/ADC1/PCINT9
25 PC2/ADC2/PCINT10
26 PC3/ADC3/PCINT11
27 PC4/ADC4/SDA/PCINT12
28 PC5/ADC5/SCL/PCINT13
1 PC6/RESET/PCINT14
ATMEGA328P
LS1
SPEAKER
R1
1.0K