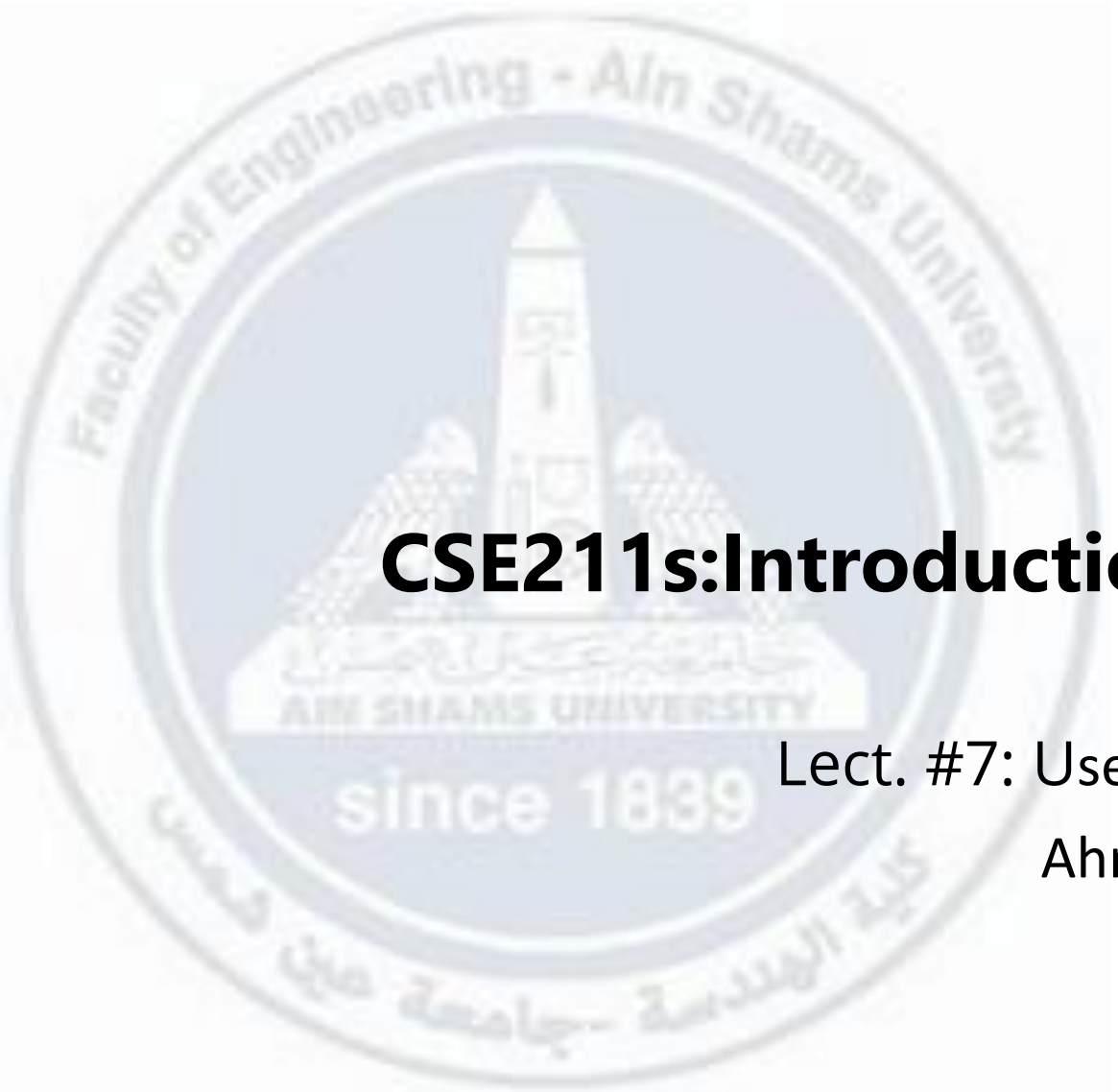


CSE211s:Introduction to Embedded Systems

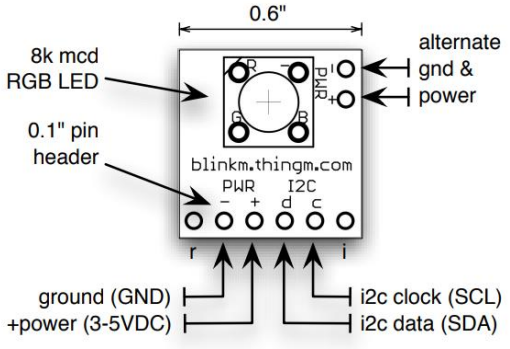
Lect. #7: Useful cases with Arduino

Ahmed M. Zaki

Controlling an RGB LED Using the BlinkM Module

BlinkM Module: Control RGB LED.
Uses I2C for communication.
Configurable I2C address with default (0x00)
Model:

BlinkM - LED07468M from
http://www.seeedstudio.com



Controlling an RGB LED Using the BlinkM Module Datasheet Example

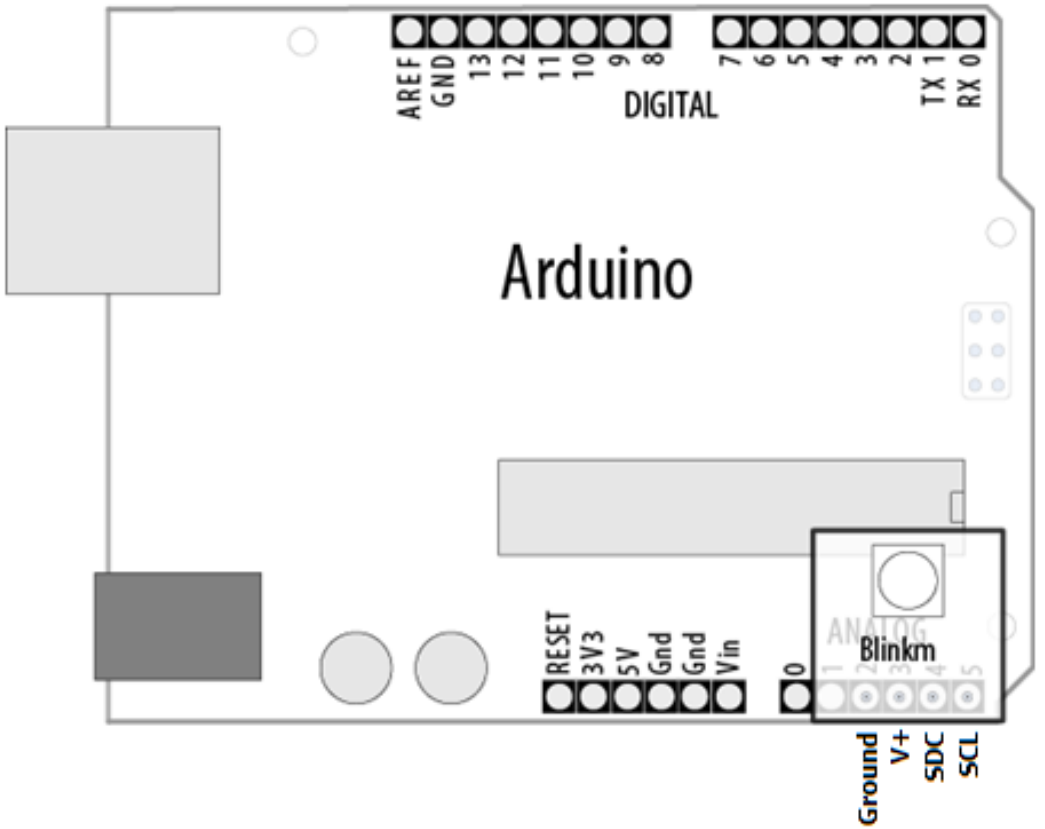
```
Wire.begin();                    // set up I2C
Wire.beginTransmission(0x09);// join I2C, talk to BlinkM 0x09
Wire.send('c');                  // 'c' == fade to color
Wire.send(0xff);                 // value for red channel
Wire.send(0xc4);                 // value for blue channel
Wire.send(0x30);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

command name	cmd char	cmd byte	# args	# ret vals	format
Go to RGB Color Now	n	0x6e	3	0	{'n',R,G,B}
Fade to RGB Color	c	0x63	3	0	{'c',R,G,B}
Fade to HSB Color	h	0x68	3	0	{'h',H,S,B}
Fade to Random RGB Color	C	0x43	3	0	{'C',R,G,B}
Fade to Random HSB Color	H	0x48	3	0	{'H',H,S,B}
Play Light Script	p	0x70	3	0	{'p',n,r,p}
Stop Script	o	0x6f	0	0	{'o'}
Set Fade Speed	f	0x66	1	0	{'f',f}
Set Time Adjust	t	0x74	1	0	{'t',t}
Get Current RGB Color	g	0x67	0	3	{'g'}
Write Script Line	W	0x57	7	0	{'W',n,p,...}
Read Script Line	R	0x52	2	5	{'R',n,p}
Set Script Length & Repeats	L	0x4c	3	0	{'L',n,l,r}
Set BlinkM Address	A	0x41	4	0	{'A',a...}
Get BlinkM Address	a	0x61	0	1	{'a'}
Get BlinkM Firmware Version	Z	0x5a	0	1	{'Z'}
Set Startup Parameters	B	0x42	5	0	{'B',m,n,r,f,t}

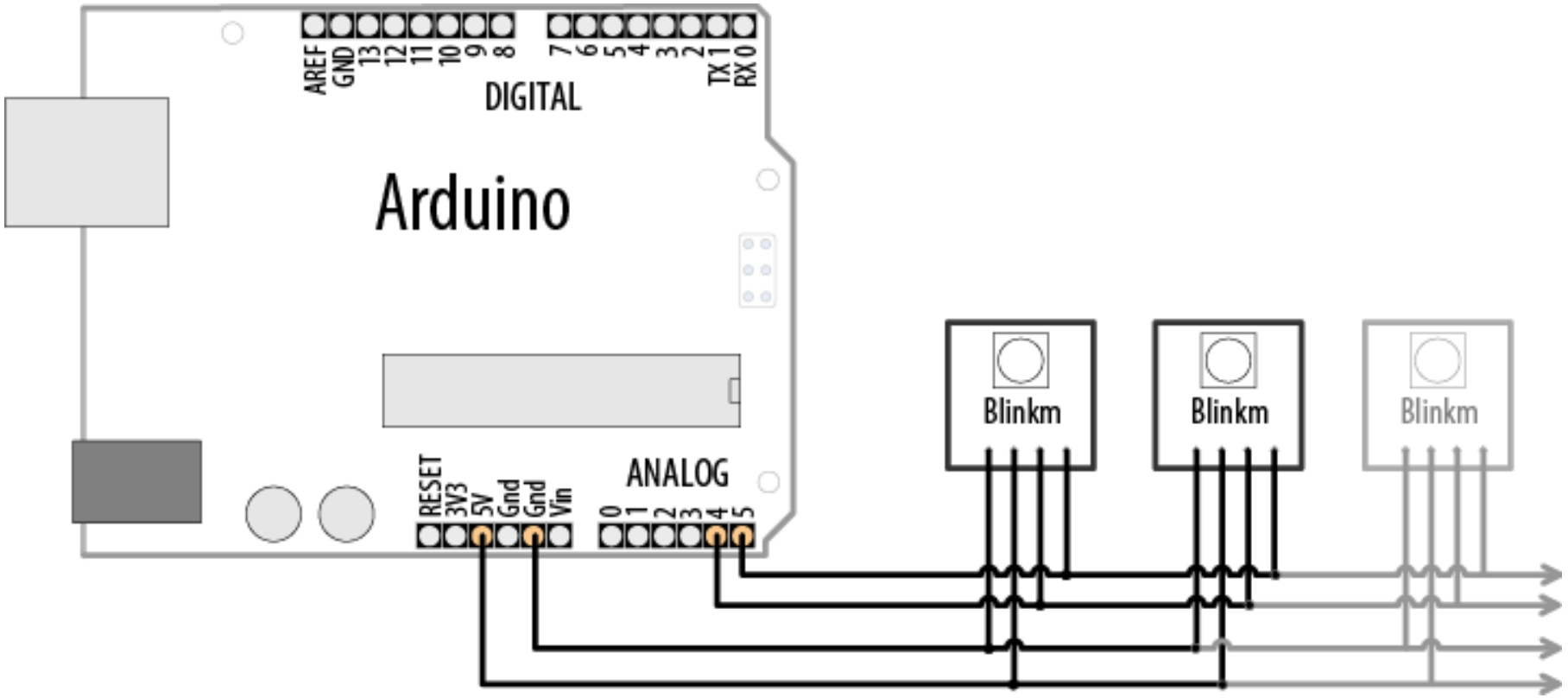
Controlling an RGB LED Using the BlinkM Module : Example

```
#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)



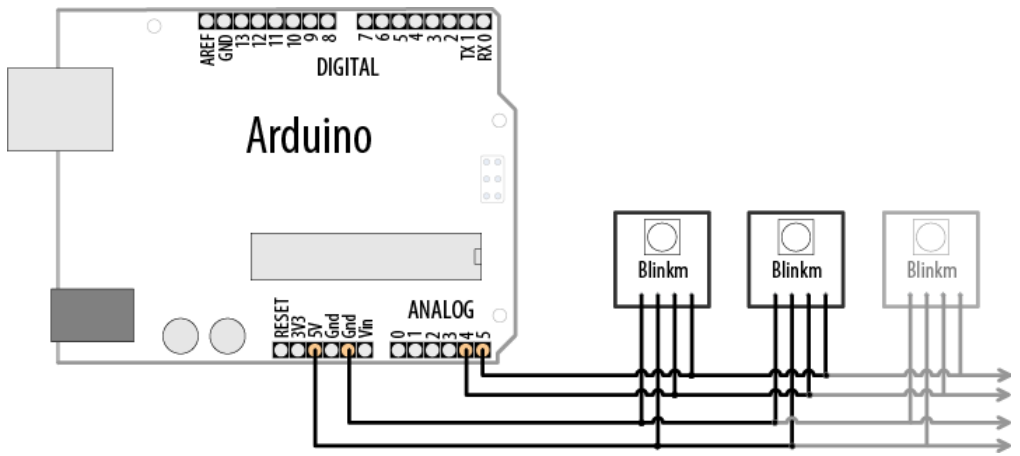
Controlling Several BlinkM of One Address



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.



Using External Real Time Clock Module

RTC Module: Produce Real Time Clock

Uses I2C for communication.

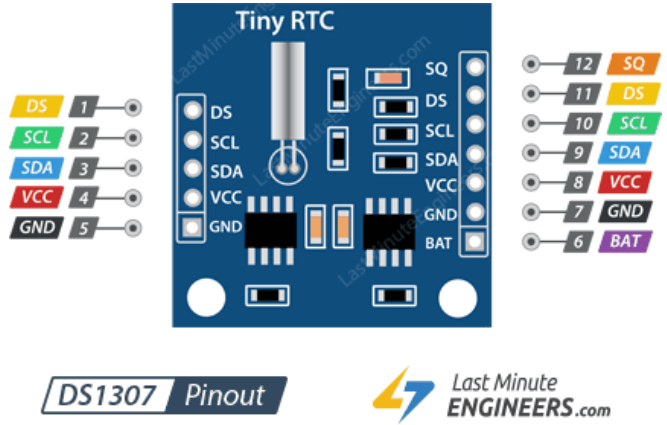
Produce 7 BCD values for (second, minute, hour, week day, day, month, year - 2000)

Example of BCD value $(0x25)_{16} \rightarrow (25)_{10}$

Configurable I2C address with default (0x68)

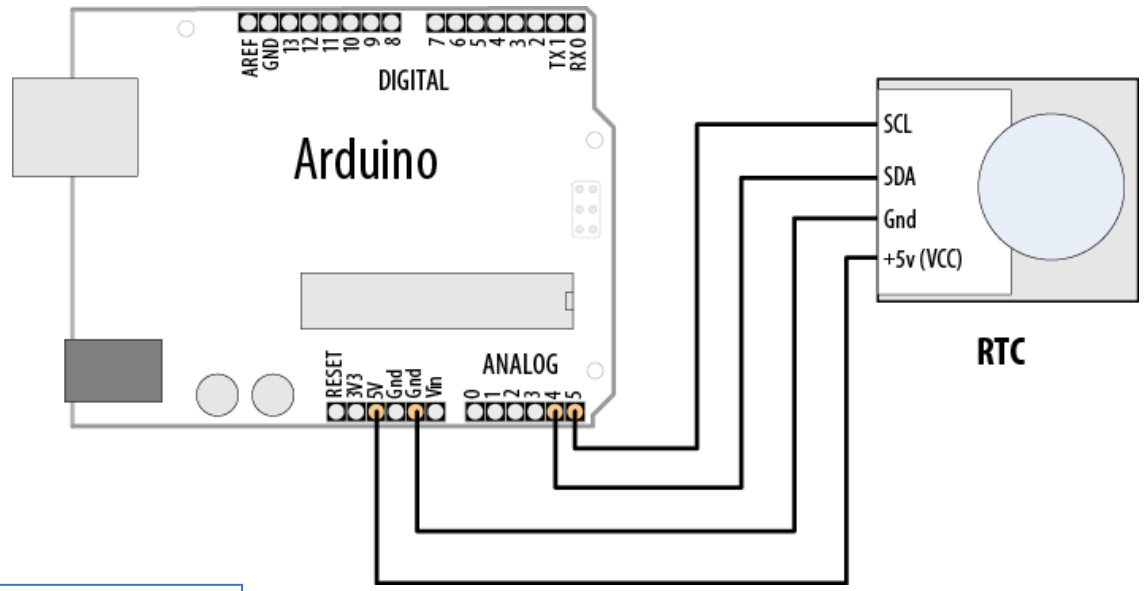
Model:

RTC- DS1307 from http://www.seeedstudio.com



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);
}
```



Wire.read()

Driving Four 7-Segment LEDs Using Only Two Wires

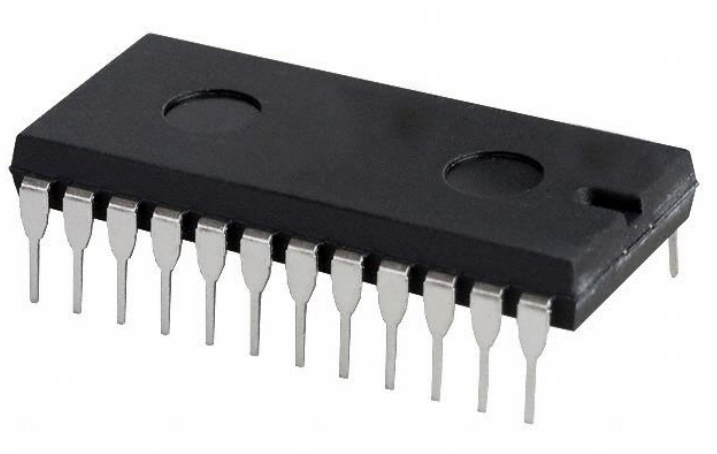
LED Driver Module

Uses I2C for communication.

Default Address: 0x38

Model:

SAA1064 from www.mouser.com



Driving Four 7-Segment LEDs Using Only Two Wires

SC	SB	SA	SUB-ADDRESS	FUNCTION
0	0	0	00	control register
0	0	1	01	digit 1
0	1	0	02	digit 2
0	1	1	03	digit 3
1	0	0	04	digit 4
1	0	1	05	reserved, not used
1	1	0	06	reserved, not used
1	1	1	07	reserved, not used

Control bits (see Fig.4)

The control bits C0 to C6 have the following meaning:

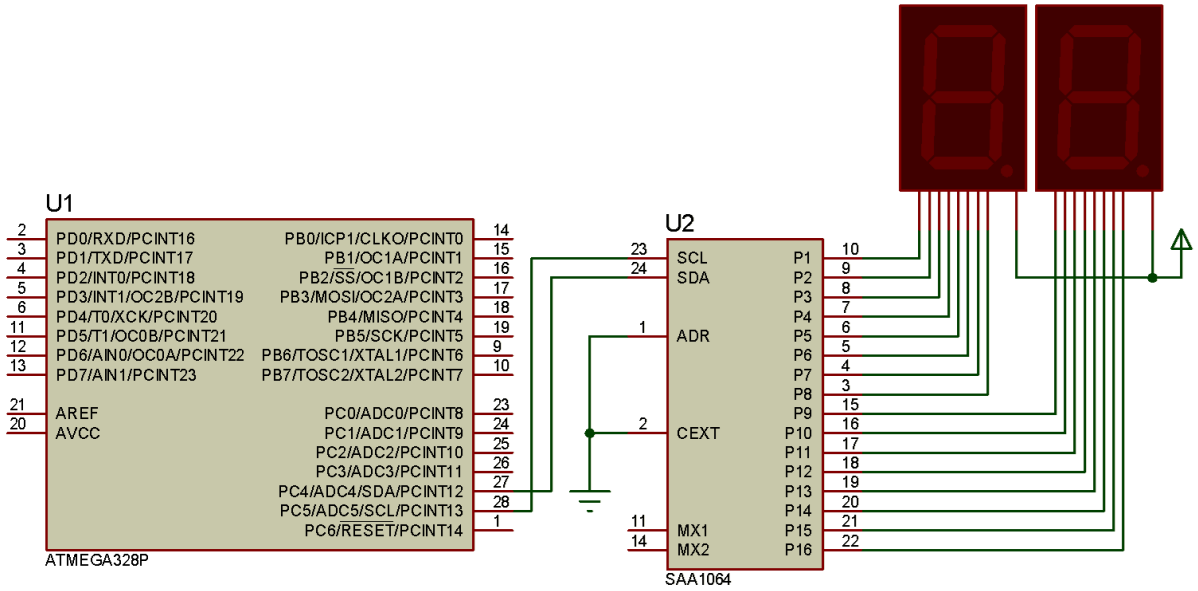
C0 = 0	static mode, i.e. continuous display of digits 1 and 2
C0 = 1	dynamic mode, i.e. alternating display of digit 1 + 3 and 2 + 4
C1 = 0/1	digits 1 + 3 are blanked/not blanked
C2 = 0/1	digits 2 + 4 are blanked/not blanked
C3 = 1	all segment outputs are switched-on for segment test(1)
C4 = 1	adds 3 mA to segment output current
C5 = 1	adds 6 mA to segment output current
C6 = 1	adds 12 mA to segment output current

Note

1. At a current determined by C4, C5 and C6.

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++;
}
```



Driving Multidigit, 7-Segment Displays Using SPI

Control 7 Segment Displays

Uses SPI for communication

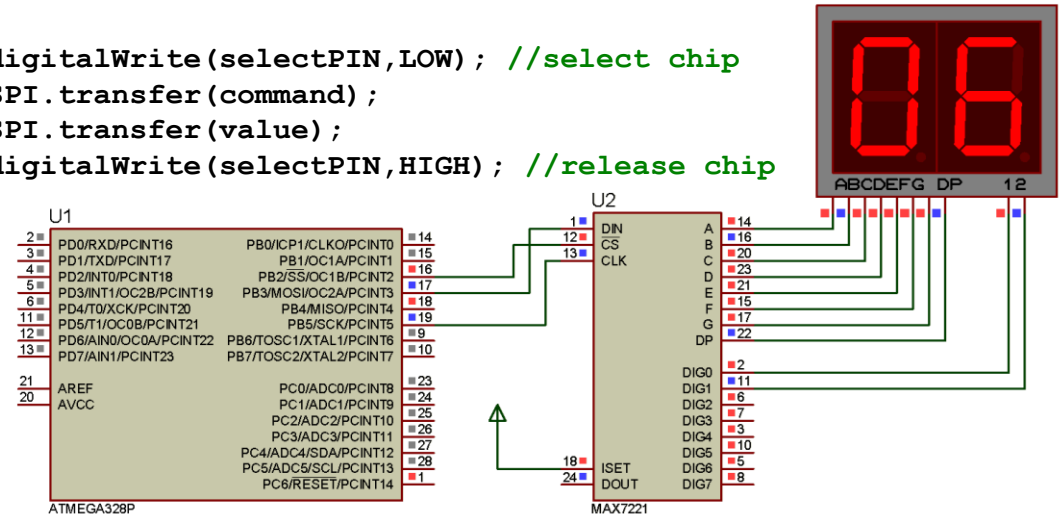
Model:

MAX7221 from www.mouser.com

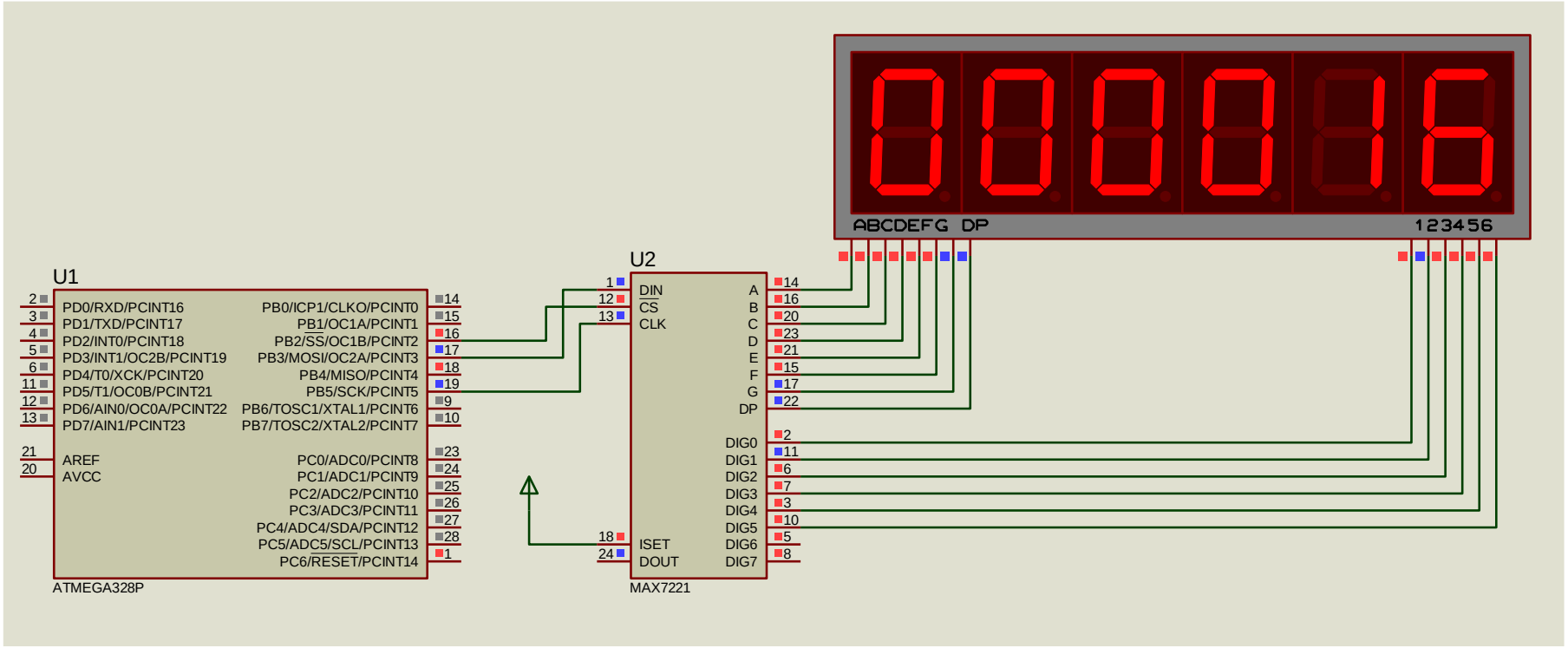
Driving Multidigit, 7-Segment Displays Using SPI

```
#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);
}
```

```
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
}
```



Driving Multidigit, 7-Segment Displays Using SPI (6 digits)



RF Communication

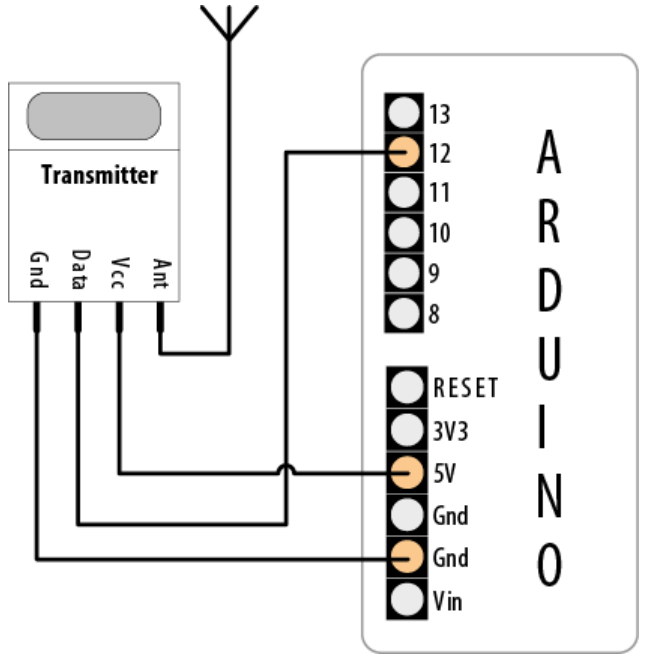
RE: Radio Frequency

Model:

- WLS107B4B (2Km with Encoder/Decoder)
- WLS102B5B (100m)
- WLS126E1P (150m)
- from http://www.seeedstudio.com

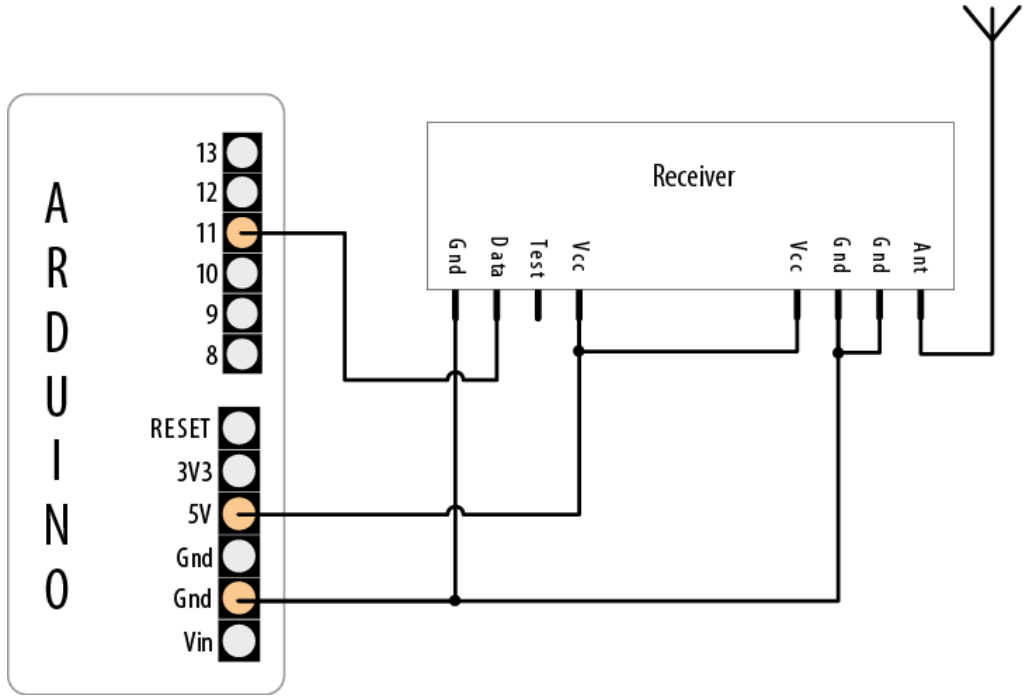
RF Communication (RF Transmitter)

```
#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)
{
      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);
      vw_rx_start();
}
void loop(){
      if (vw_get_message(message, &msgLength)){
            Serial.print("Got: ");
            for (int i = 0; i < msgLength; i++)
                  Serial.write(message[i]);
            Serial.println();
      }
}
```



Display Devices

LCD Text Display
LCD Graphics Display
TV Interface

LCD Text Display

LCD: Liquid Crystal Display

Uses industry standard HD44780

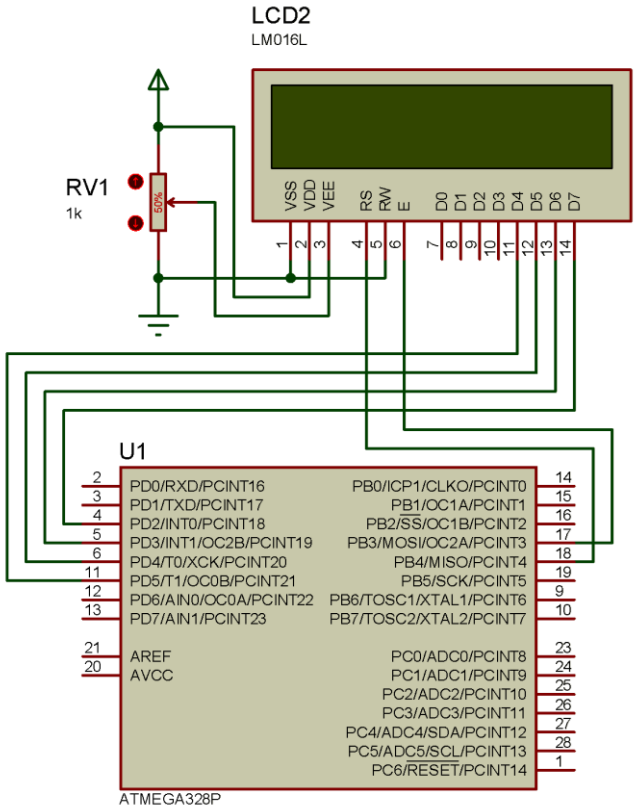
Uses serial communication

Models

LCD1602 (LCD23154P) from http://www.seeedstudio.com

LCD Text Display Example

```
#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);
}
```



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);
     }
}
```

LCD Text Display: Displaying Special Symbols

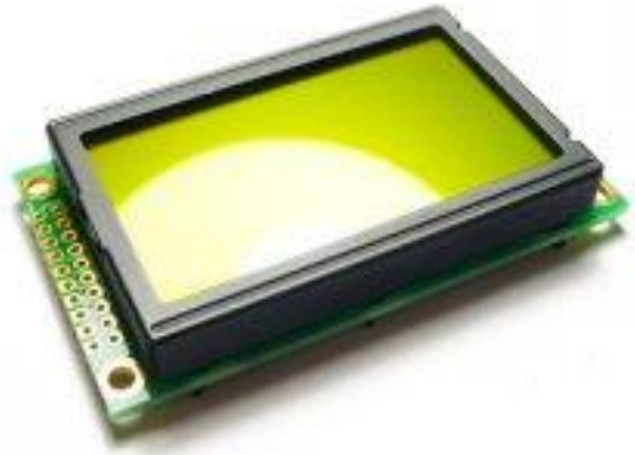
```
#include <LiquidCrystal.h>
const int numRows = 2;
const int numCols = 16;
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
void setup()
{
     lcd.begin(numRows, numCols);
     showSymbol(B11011111, "degrees");
     showSymbol (B11110111, "pi");
     showSymbol(B11101100, "cents");
     showSymbol(B11101000, "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);
}
```

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);
     }
}
```

LCD Graphics Display

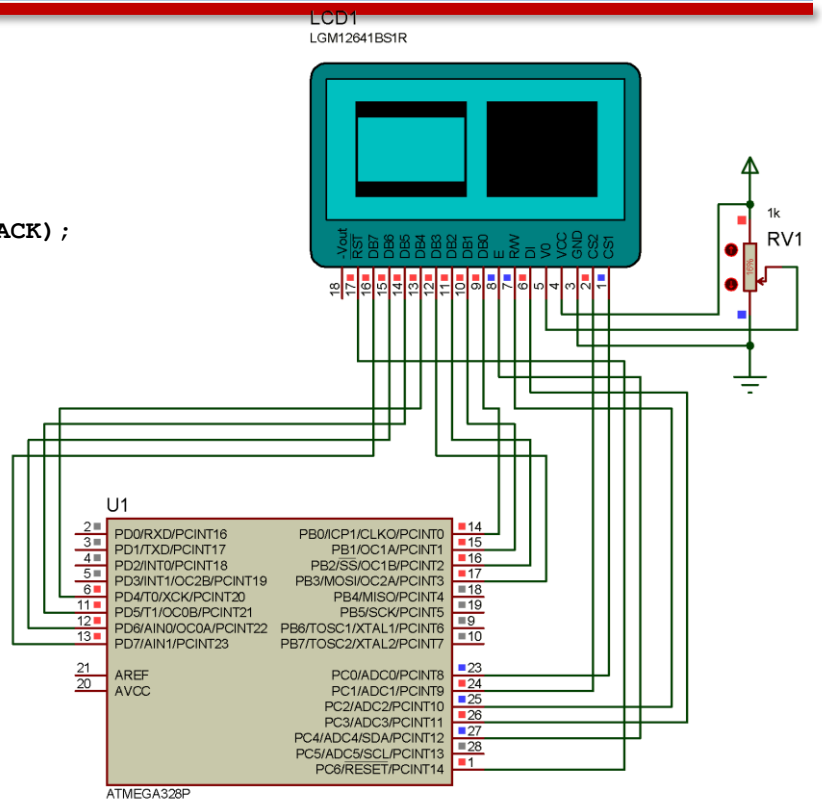
- LCD Graphics: Liquid Crystal Display with Graphics Support
- Uses industry standard KS0108
- Uses serial communication
- Models
  - LCD102B6B from http://www.seeedstudio.com



LCD Graphics Display

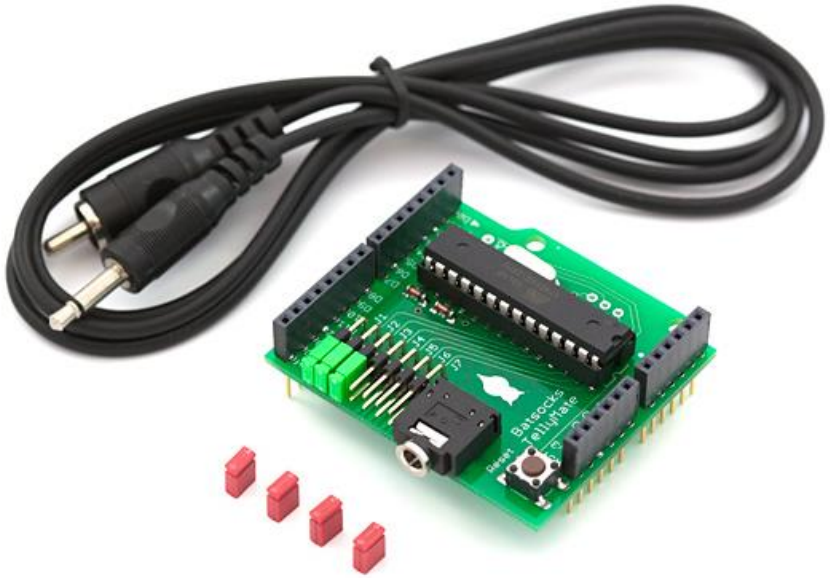
```
#include <ks0108.h>
#include <Arial14.h>
#include "SystemFont5x7.h"
#include "ArduinoIcon.h"
#define SIMFACT 10
unsigned long startMillis;
unsigned int iter = 0;
void setup(){
     GLCD.Init(NON_INVERTED);
     GLCD.ClearScreen();
     GLCD.DrawBitmap(ArduinoIcon, 32,0, BLACK);
     delay(1000/SIMFACT);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



TV Interface

- Produce Analog Video Signal to TV
- Controlled by Serial Interface
- Model
  - TellyMate DEV-09313 from www.sparkfun.com



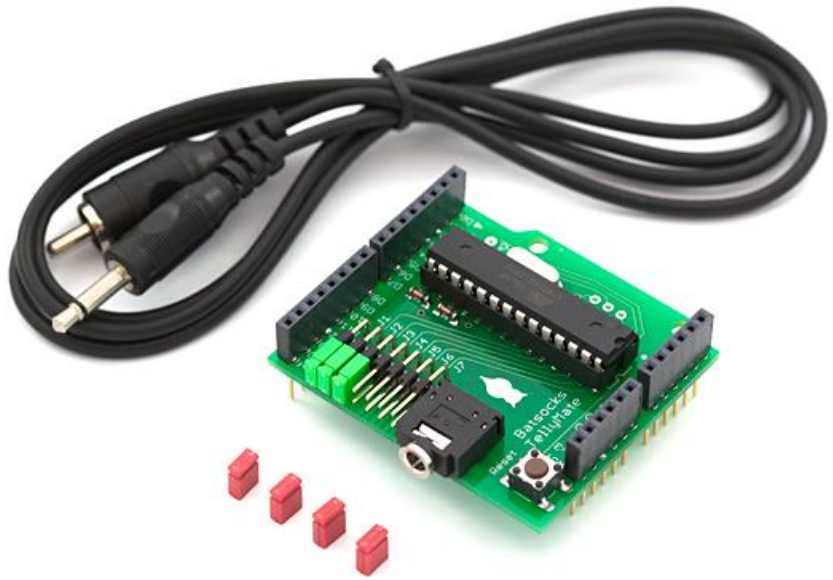
TV Interface

The TellyMate Shield equips an Arduino with the ability to send simple text and graphics to a TV. The TellyMate Shield connects to an Arduino board using long headers which extend through the shield.

This keeps the pin layout intact and allows another shield to be stacked on top.
Arduino uses digital pin1 (TX) to communicate with the TellyMate.

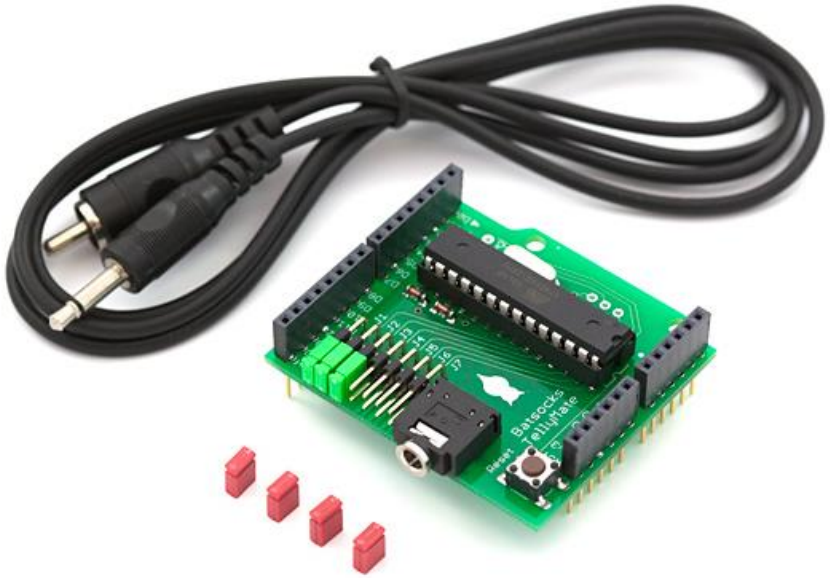
Just plug it into your Arduino and use Serial.println() commands to output text onto your TV!

The shield provides a 3.5mm jack and the necessary cable to convert to a composite video connection. The baud rate is selected by jumpers, which are also included.



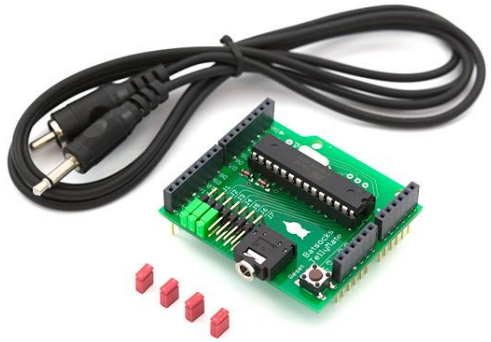
TV Interface

- TV output from your Arduino
- PAL or NTSC Composite Video
- Stackable Shield
- works with Serial.println() etc.
- 38x25 characters
- Black and White
- Simple Graphics
- Double width / height text
- Simple control codes



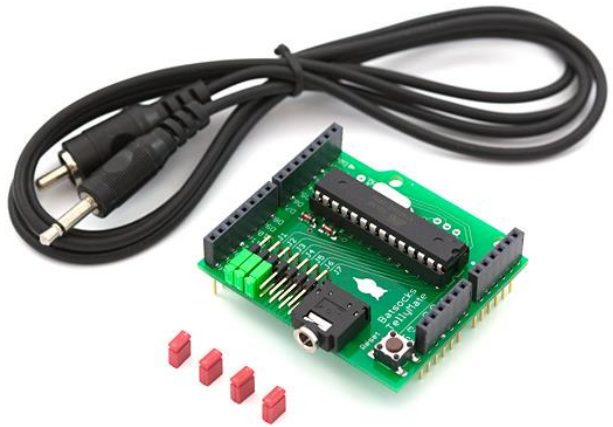
TV Interface

Mnemonic	Code (Hex)	Name	Comments
<NUL>	00	Null	Never affects the device. <NUL> codes are *always* ignored wherever they appear in the character stream.
<BEL>	07	Bell	Not currently implemented
<BS>	08	Backspace	Moves the cursor one position to the left. No characters are overwritten. If the cursor is already at column 0, there is no effect.
<TAB>	09	Tab	Moves the cursor to the next tab position. Tab positions are every 4 columns (0, 3, 7 etc.). If the cursor is already at a tab position, it is moved to the next. If the cursor is beyond the last tab position (column 35), there is no effect.
<LF>	0A	Linefeed	cursor down one row, scrolling the screen if it's already on the last row. The cursor's column is unchanged unless the Auto CR option is set.
<FF>	0C	Formfeed	This clears the screen and places the cursor in column 0 of the bottom row.
<CR>	0D	Carriage Return	This moves the cursor to column 0 of the current row. No Linefeed occurs unless the Auto LF option is set.
<DLE>	10	Data Link Escape	This code supresses any interpretation of the next character. The next character's glyph is output to the screen, regardless of its normal meaning. *Note: The <NUL> is an exception, which is always ignored.* This code is used to output otherwise non-printable glyphs to the screen.
<CAN>	18	Cancel	This code immediately cancels any current escape sequence. Any already-processed effect of that escape sequence will remain (see <ESC>Y for an example).
<ESC>	1B	Escape	This code initiates an escape sequence. See the 'Escape Sequences' table, below for further details.



TV Interface

Sequence	Parameter(s)	Standard	Name	Comments
<ESC>A		VT52	Cursor Up	Moves the cursor up. If the cursor is already on the top line, there is no effect.
<ESC>B		VT52	Cursor Down	Moves the cursor down. If the cursor is already on the bottom line, there is no effect.
<ESC>C		VT52	Cursor Right	Moves the cursor right. If the cursor is already at the right-hand-side, there is no effect.
<ESC>D		VT52	Cursor Left	Moves the cursor left. If the cursor is already at the left-hand-side, there is no effect.
<ESC>E		H19	Clear Screen and Home	Home is row 0, column 0.
<ESC>H		VT52	Cursor Home	Home is row 0, column 0. Nothing is cleared.
<ESC>I		VT52	Reverse Line Feed	Also known as 'Reverse Index'
<ESC>J		VT52	Clear to end-of-screen	Clears from the cursor (inclusive) to the end of the screen. The cursor is not moved.
<ESC>K		VT52	Clear to end-of-line	Clears from the cursor (inclusive) to the end of the line. The cursor is not moved.
<ESC>Q		TellyMate	Diagnostic Information	Clears the screen and outputs diagnostic details about the terminal. This sequence takes a few tens of scanlines to complete. This may cause a brief glitch in the video output. A delay of 2ms should be observed before further characters are sent. *Note: The 'Serial: jumpers' line shows the current state of the jumpers. There may be discrepancies between the baud rate selected by jumpers and the actual baud rate being used. This is because the 16Mhz clock cannot be accurately divided to the requested speed.*
<ESC>Y	rc	VT52	Direct Cursor Addressing	Moves the cursor to the specified row and column. r is the ascii character of 32 + the desired row. c is the ascii character of 32 + the desired column. e.g. to position the cursor at row 5, column 7, ascii characters 37 and 39 should be sent (% and '). to position the cursor at row 0, column 0, two ascii characters 32 (<SPACE>) should be sent. If this sequence is cancelled (by receipt of a <CAN> code) after the row parameter has been received, it's effect remains - Cancelling the sequence does not undo any actions already completed. This method can be used to solely set the cursor's row. If a supplied parameter would lead to an off-the-screen cursor position, then that parameter is quietly ignored. This feature can be used to change just the cursor's row or column.



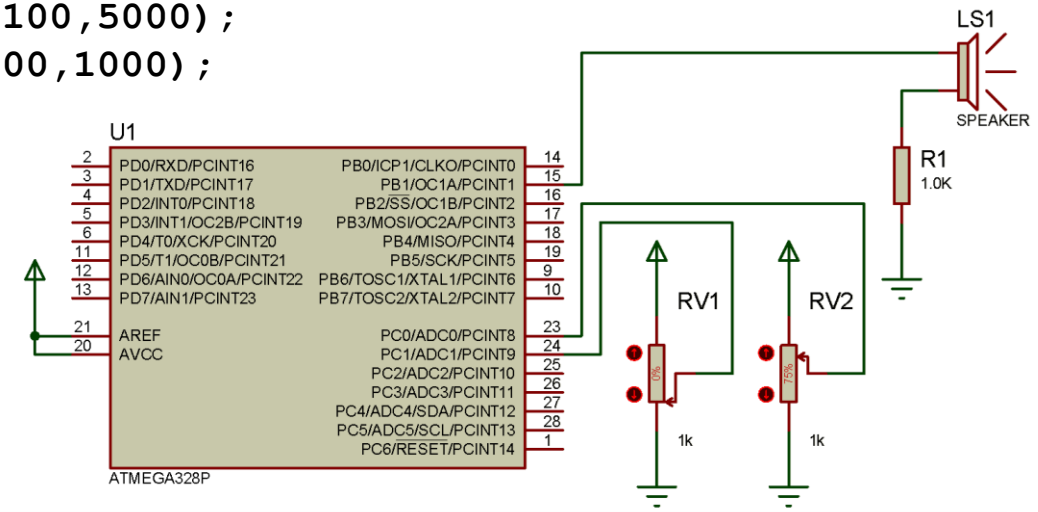
TV Interface

```
const byte ESC = 0x1B;
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)) ;
}
```

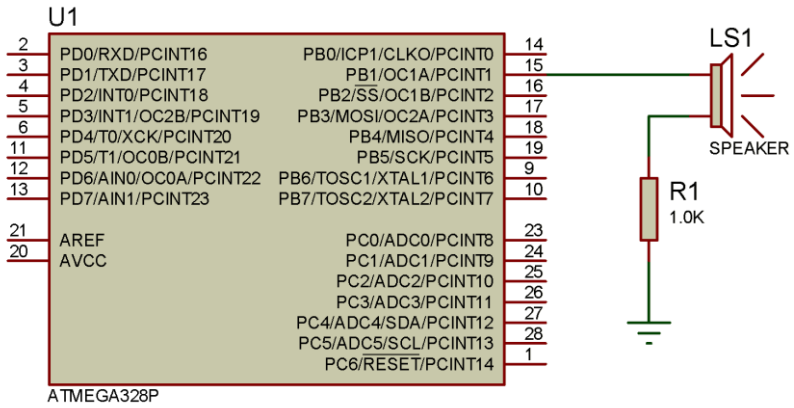
Playing Tones

```
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);
}
```



Playing a Simple Melody

```
#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);
}
```



References

1) https://static1.squarespace.com/static/5c155684f407b4100552994c/t/5c2d20ca0e2e7292108eadf8/1546461407535/BlinkM_datasheet.pdf
2) https://lastminuteengineers.com/ds1307-rtc-arduino-tutorial/
3) https://datasheet.octopart.com/SAA1064T-N2,112-NXP-Semiconductors-datasheet-17703633.pdf