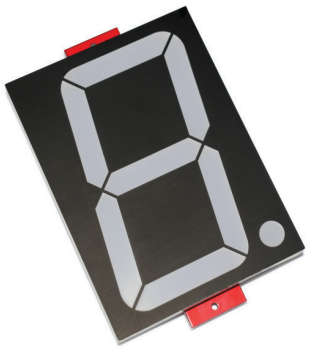
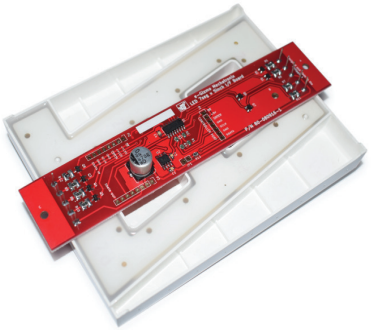
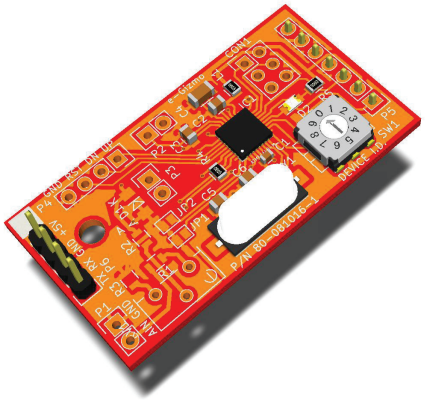
LED 7segment x 5inch I/F Board

Technical Manual rev1.0







Seven segments LED display units posses some qualities that are not easily replaced by newer display systems (like LCDs), hence, continues to be popular even with the proliferation of cheaper LCD based systems. They are usually more visible especially when viewed at a distance, oftentimes more pleasing to look at, more durable and lasts longer - that just mentioning a few.

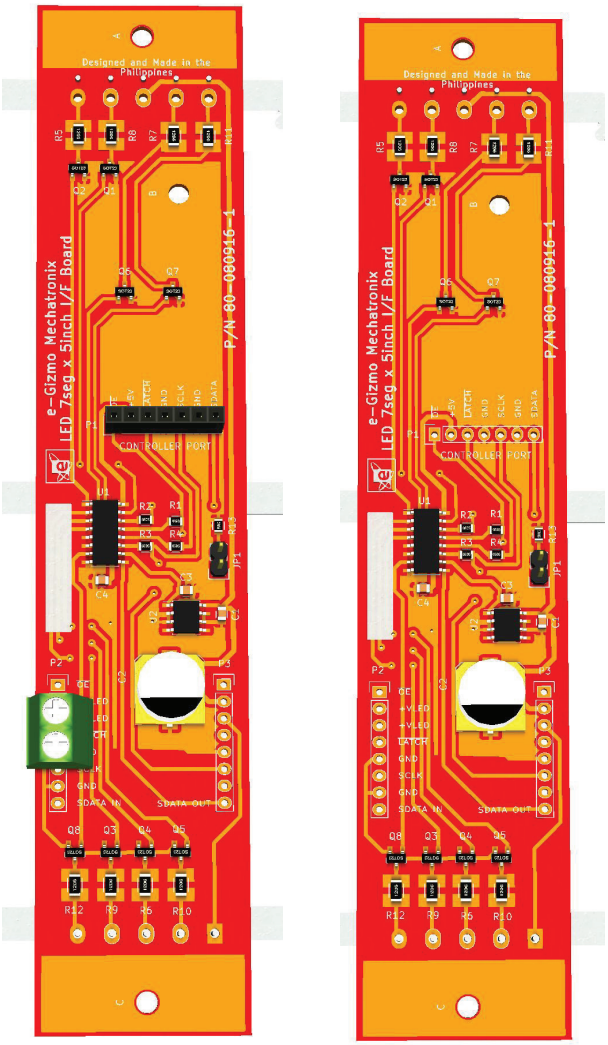
But to a maker, assembling and programming one can be an exacting one.

The Serial LED Display System kit is design to make integration of LED display to your circuit as painless as possible. Instead of long codes, you just need to send simple serial commands to display some numbers.

The Serial LED Display system comprises of a LED controller module and the 7 segment LED display panels. Each LED controller module can drive up to 16 LED display panels. The command set allows you to display a group of digit in any arbitrary position, giving you the freedom to break (up to 16 digits) display into smaller subgroup.

The LED controller is addressable. This allows you to connect multiple Serial LED Display over a single UART channel.

The LED display panels, as of this time of writting, is available in 5-inch size only. Other sizes will be made available very soon.



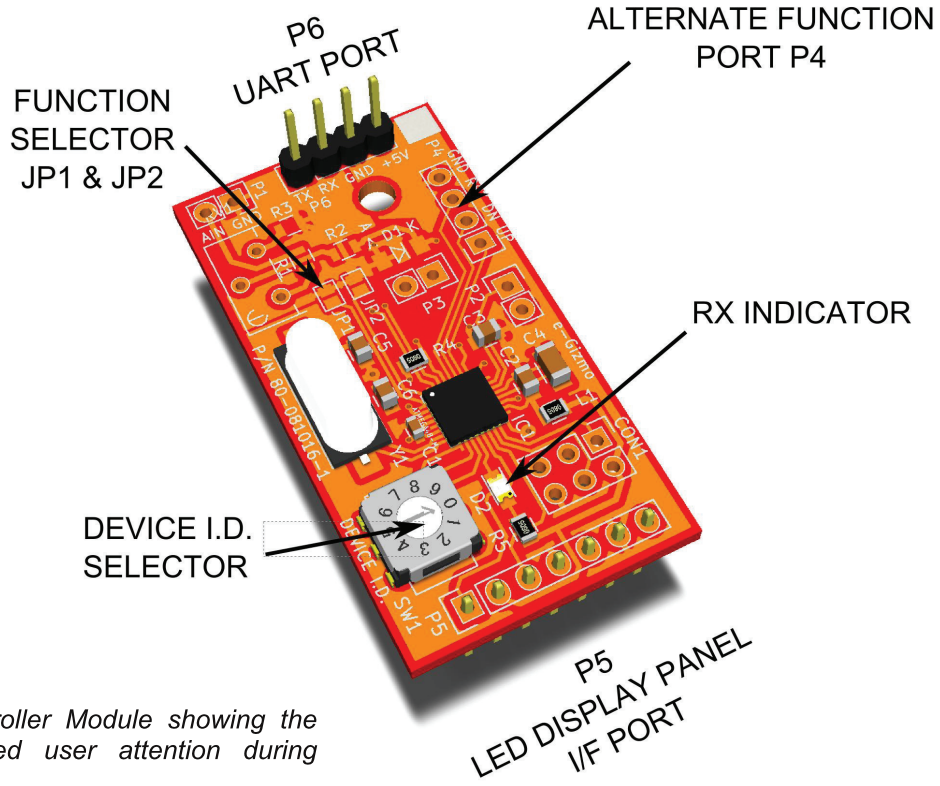


Figure 2. LED Controller Module showing the parts that may need user attention during installation.

SW1 - DEVICE ID

The Device ID switch assigns a specific ID for this device. Device ID allows the controller to selectively accepts commands over a shared UART channel. Gently rotate using a small flat screwdriver until the arrow end of the adjustment slot coincides with the desired device ID.

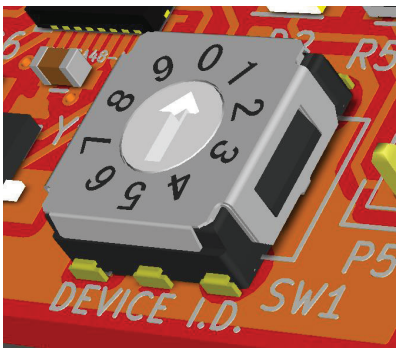


Figure 3. The LED Controller Module will process commands only when the command packet ID matches with its setting.

ALTERNATE FUNCTIONS

The Serial LED Controller has two alternate function: It can be configured to work as a stand alone UP/DOWN counter or a Stop Watch. This can be effected by shorting JP1 or JP2 accordingly, as summarized in Table 1.

These functions requires an installation with 5 panels (5 digit system). These also requires three push button switches to be installed on port P4. Please refer to the wirings section for more details.

As a counter, it can count UP or Down within a range of 0 to 32,768.
As a stop watch, it can display time of up to 59 minutes and 59 secconds long with a display resoulution of 0.1s

Important: Timer accuracy is determined by the crystal used in the system, hence, is probably not as accurate as a proper stopwatch.

Table 1. JP1 & JP2 Function Selector

The Serial LED controller has two alternate stand alone functions: An up/down counter and a Stop watch. See text for a more detailed description.

JP1	JP2	FUNCTION
0	0	Serial LED with CRC check
1	0	UP/DOWN Counter
0	1	Stop Watch
1	1	Serial LED (CRC checking OFF)

Table 2. P6- UART PORT

The UART port is the serial LED display system gateway to the host. You can connect your host controller UART directly, or wirelessly via a wireless transparent uart device (e.g. e-Gizmo UHF Data Transceiver or HM-10 Bluettoth Module).

PIN	ID	DESCRIPTION	
1	+5V	Power +5V	P
2	GND	GND	P
3	RXD	Receive Data	I
4	TXD	Not Used	X

Table 3. P4- STOP WATCH/COUNTER

Push button switch port when configured as Counter or Stop Watch.

PIN	ID	DESCRIPTION	
1	UP	Count UP/Timer Start	I
2	DN	Count DN/Timer Stop	I
3	RST	RESET	I
4	GND	GND	P

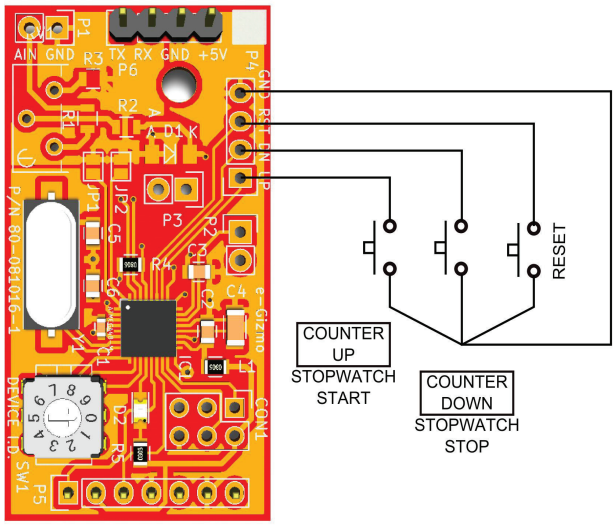
Table 4. P5- LED Display Panel I/F Port

This mates to the seven segment LED display panels designed for this system.

PIN	ID	DESCRIPTION	
1	~OE	Output Enable	O
2	+5V	Power IN +5V	P
3	LATCH	Latch	O
4	GND	GND	P
5	SCLK	Serial Clock	O
6	GND	GND	P
7	SDATA	Serial Data	O

Legend:

I - Input
O- Output
P- Power
X - not used



Alternate Function using LED controller module for counter UP/DOWN.

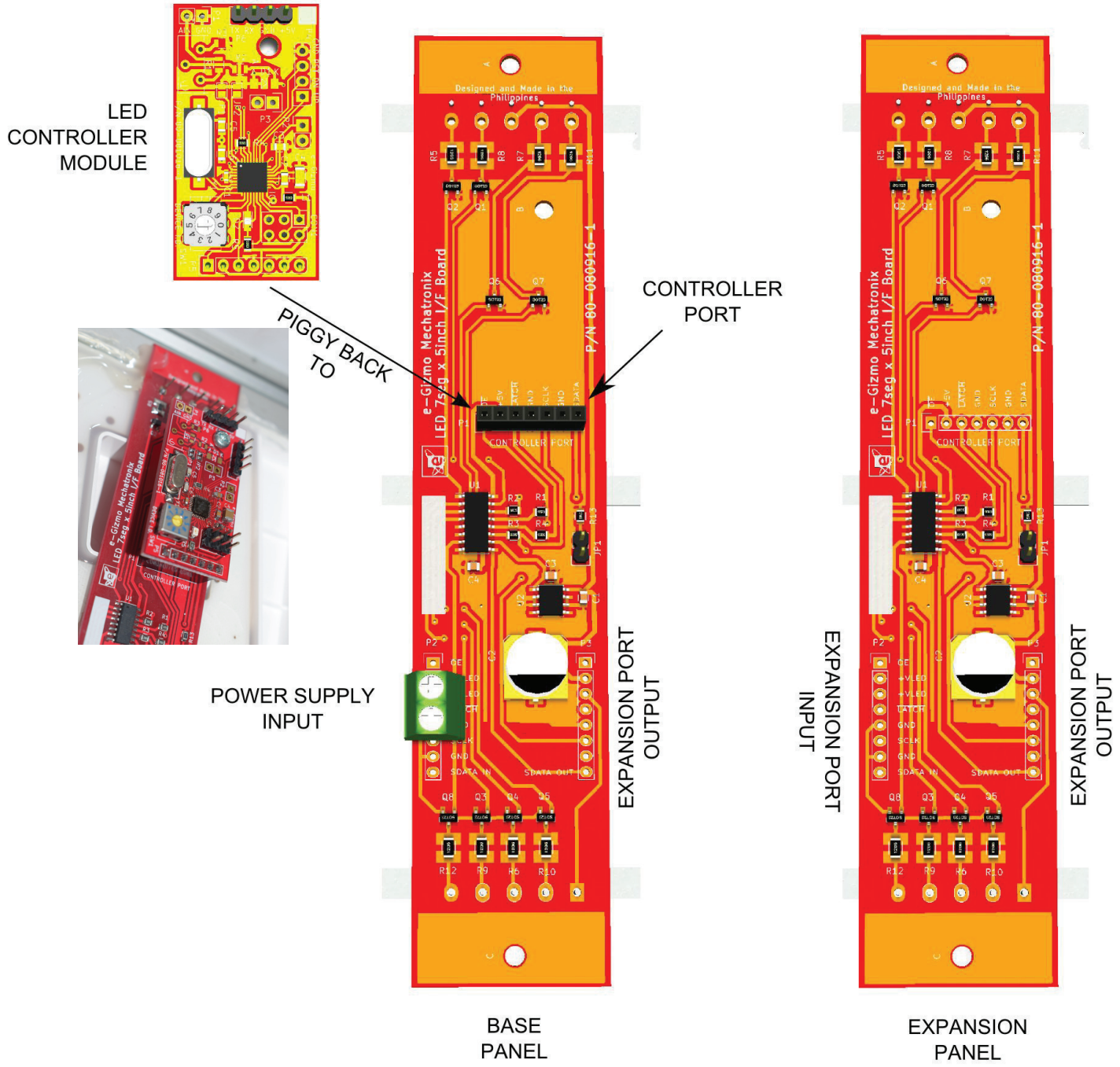


Figure 4. The Serial LED requires at least one Base Display panel. This is where the LED Controller Module is piggy back installed and the power supply is wired. The Base Panel + Controller combination acts as the master Serial LED Module. You can connect additional expansion panels (up to a total of 16 panels) as your application requires.

LED PANEL I/F BOARD

The LED Panel I/F circuit is based on a 74HC595 serial to parallel chip, hence the data interface lines work exactly as described in the chip datasheet. Please refer to these datasheets if you want to study the workings in greater detail.

The LED controller board connects through this port and takes care of the operation with the LED Panels. Generally, you do **not** have to dig into the details when used with the LED controller.

Base and Extension Panel

The base and extension panel share the same circuitry but are built a bit differently. Tbe base panel has a socket strip installed (controller port) for easy insatllation of the LED Controller Module. It also has a terminal block installed in the expansion port (input side) for the system power supply input. (See figure 4)

A typical installation requires just one base panel, and from the base panel, up to 15 additional extension panels can be daisy chained connected.

Table 6. P1-LED Display Panel I/F Port

Installation slot for the LED Controller Module.

PIN	ID	DESCRIPTION	
1	~OE	Output Enable	I
2	+5V	Power OUT +5V	P
3	LATCH	Latch	I
4	GND	GND	P
5	SCLK	Serial Clock	I
6	GND	GND	P
7	SDATA	Serial Data	I

Table 7. P2- Expansion Port Input Side

PIN	ID	DESCRIPTION	
1	~OE	Output Enable	I
2	+VLED	LED + SUPPLY	P
3	+VLED	LED + SUPPLY	P
4	LATCH	Latch	I
5	GND	GND	P
6	SCLK	Serial Clock	I
7	GND	GND	P
8	SDATA	Serial Data In	I

Table 8. P3-Expansion Port Output Side

PIN	DESCRIPTION	
1	Output Enable	I
2	LED + SUPPLY	P
3	LED + SUPPLY	P
4	Latch	I
5	GND	P
6	Serial Clock	I
7	GND	P
8	Serial Data Out	O

Legend:

I - Input
O- Output
P- Power
X - not used

Construct this diagram Figure 5 (with base panel) and open the ***Serial LED arduino sample.ino***

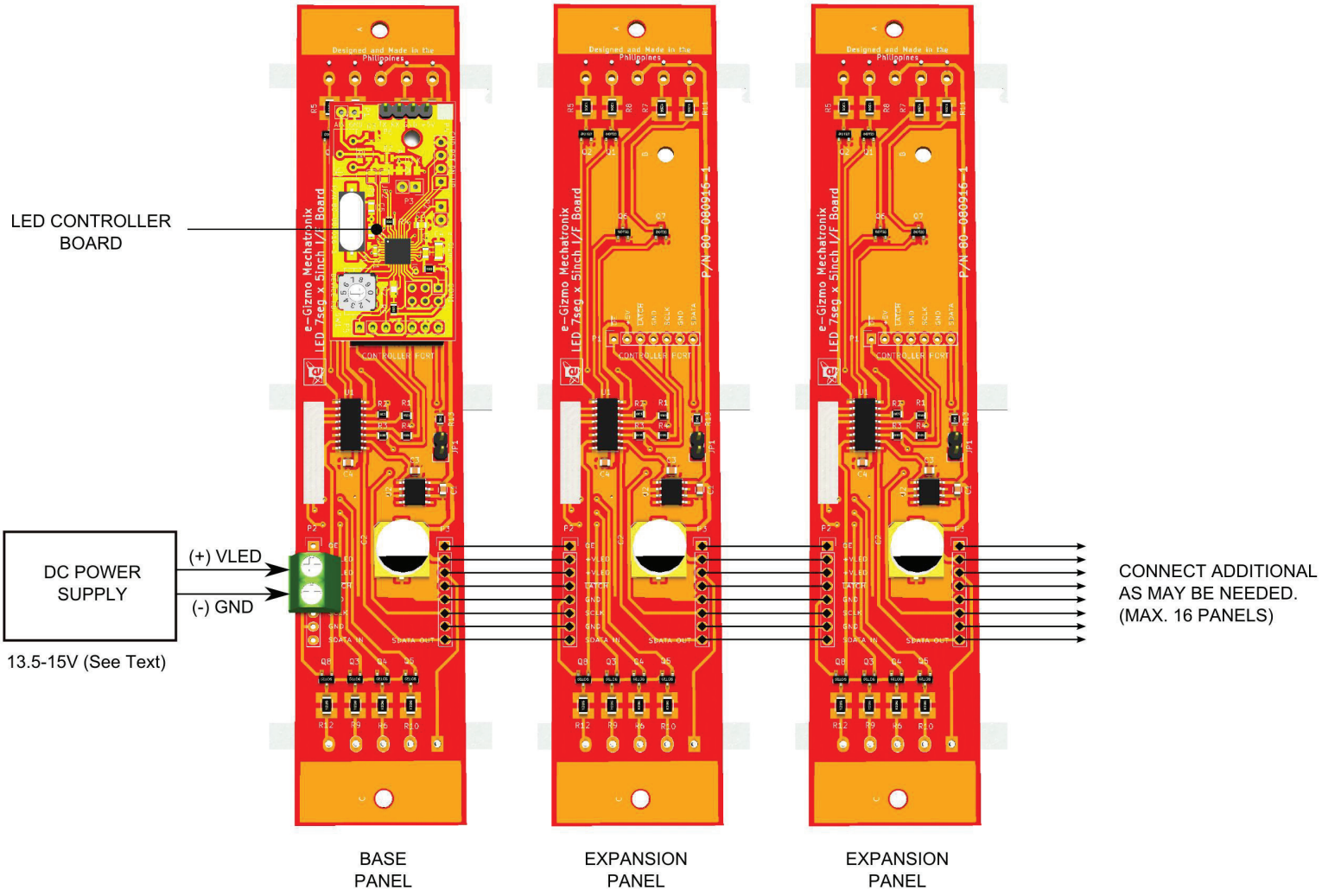


Figure 5. Serial LED Display system wiring example. The last (left most) module in the chain is assigned digit No. 1, and increment accordingly as you point towards the base panel.

Figure 6. Wiring example of a Serial LED Display Unit in a installation. PFC and ribbon wire connectors are used to connect the modules together. This results in a easy and neat wiring installation.

APPLICATION NOTES

Power Supply Requirements

1. The LED Controller Module gets its +5V supply from the base panel.
2. The LED Panels works best with a DC supply voltage of 14VDC. Use a well regulated DC source. Each panel consumes about 70mA with all segments ON. This increases to 100mA per panel at 15VDC. Hence, to determine the power supply current requirement, simply multiply the current by the number of panels installed.

Example: System with 16 panels (16 digits display)

@14VDC, the system will draw 16* 0.07A = 1.12A with all the LED segments driven ON. You should use a power supply with an ampere greater than 1.12A. A 14V, 1.5Amp power supply will be a good choice.

Connecting a host

The host controller (your own choice of controller, like Arduino) connects with the LED Display System (LED controller) via their own respective UART port. The important things to remember are:

1. The TX output of the host should connect to the RX input of the LED Controller Module.
2. Keep the connection short. If wiring exceeds, say 10cm, twist the wire together (RX-TX and GND) along its length.
3. For installations that requires really long connections between the host and display, use of RS-422/485 board in between will be necessary. These interface modules will allow you install wire runs as long as 4000 meters measured from end to end with reasonable data transmission reliability. Wireless link using data transceivers is also an option, a wireless setup will be beset with frequent drops in transmission, make sure your codes are up to task and your application can tolerate these data losses.

COMMUNICATIONS MANUAL

Baud Rate: 9600
Data: 8 Bit
Parity: none
Handshake: none
Device ID: As per user setting

Device ID:

The LED display controller module is addressable. It requires you to supply a single digit address, and only responds to serial commands that matches its set address. This allows you connect mutiple devices using just one UART port in the host controller. This also allows simpler wireless setup with the use of transparent uart wireless modules.

Cyclic Redundancy Check CRC8 Byte:

The CRC byte provides a means for the LED display controller to check the integrity of the received data. If enabled, the controller expects the host to send a CRC byte immediately preceeding the [ETX] marker. It does its own calculation based on the data it received and compares it with the one transmitted by the host. If a mismatch occurs, data transmission error is detected and the led controller will discard the data packet.

CRC checking is enabled by default. It can be disabled by shorting JP1 and JP2 (Table 1), but this is not recommended. CRC checking will make coding a bit more difficult, but the added effort will be well worth it. Once enabled, the host should append a crc byte at the end of every packet to make a succesful data transmission.

Summary of Functions

_CRC - Enable CRC checking
D - Display digits
P - Configure number of panels (digits) installed
T - Test ON all display segments
C - Clear Display
c - Clear specified region of display

Communications Format

Every packet of data transmission are wrapped inside an [STX] and [ETX] marker. The device address (in ASCII format) should immediately follow after the [STX] marker.

```
[STX] – Start of transmission marker,
ASCII value  = 0x02

[ETX] – End of transmission marker, ASCII
value  = 0x03
```

The first character after the [STX] marker and device address is a single character function specifier. Each transmission may contain just a function specifier only, or may contain a series of data in addition to the function specifier. End of transmission is indicated by the [ETX] marker.

[STX] and [ETX] are data packet markers and should not be transmitted as literal string. They should be send in their ASCII representation. The correct way of transmitting the [STX] and [ETX] markers are as shown in the following example:

Example 1: Test Segment (no crc checking)

```
Transmission Format: [STX]+5+T+[ETX]
```

This should be transmitted in their ASCII code representation as shown in the following table:

Symbol	STX	5	R	ETX
Hex	0x02	0x35	0x52	0x03

Visual Basic:

Correct:

```
'correct way to send [STX] & ETX marker

Serial1.print(chr(2)+"5T"+chr(3))
```

Wrong:

```
Serial1.print("[STX]5R[ETX]") 'WRONG!
```

Arduino:

Correct:

```
// correct way to send [STX]
Serial.write(0x02);
// Address=5, T = test command
Serial.print("5T");
// [ETX] marker
Serial.write(0x03);
```

Wrong:

```
Serial.print("[STX]5T[ETX]");   // WRONG!
```

Alternately, you can use the C/Arduino "\" operator to send the ASCII code of STX and ETX, together with the function and data:

```
Serial.print("\0025T\003");
// where "\002" = STX, "\003" = ETX
```

Notice that in the example, only the STX and ETX marker need to be manually converted to their ASCII code, for the simple reason that they have no equivalent printable characters. The three line implementation (long format)may make your program longer, but is more human readable. Hence, for clarity, all example codes given are shown in the long format. We leave it up to you if you want to convert and code it in short format.

Function Description

note: discard crc if CRC checking is not enabled.

1. **T –** Test Segments

This turns ON all LED segments for about 2 secs. Use it to check visually the 7-segments display condition.

Format: [STX]+ deviceid + T + crc + [ETX]

Where:
d - device id
c - crc8 checksum (only if crc checking is enabled)

2. **_CRC** - Enable crc8 checking

You can enable CRC checking for LED controllers with CRC checking disabled by jumpers through this command. Once enabled, crc checking can only be disabled again by power cycling (power off and then on again the led controller).

Format: [STX]+ deviceid + _CRC + [ETX]

3. **P** - Set number of installed display panels.

The led display controller, as delivered, is configured for 16 digits display (maximum number of panels a single LED controller can drive). If your installation has different number of panels installed, you should sent this command during your system initialization.

Note: The number of installed panels is saved in internal EERAM and is automatically loaded the next time you power up the led display controller. But it is highly recommended to include this command in your system initialization codes.

Format:[STX] + deviceid + P + panels + crc + [ETX]

If you put in a value that is less than what is actually installed, the display will be duplicated on the "excess" panels.

4. **D** - Display digits

Display the digits of specified size starting in the specified position.

Format:[STX]+ deviceid + D + start + size + todisplay + crc + [ETX]

where:
start - start displaying in this position
(0 to F, 0 = 16)
size - how many digits it will occupy
(0 to F, 0 = 16)
todisplay - number string to display

note: start and size field is a single character hex digit representation, with 0 representing the equivalent of 16.

5. **C** - Clear Display

Clear all display.

Format:[STX] + deviceid + C + crc + [ETX]

6. **c** - Clear a Region in Display

Clear a portion of specified size and location.

Format:[STX]+ deviceid + c + start + size + crc + [ETX]

where:
start - start clearing in this position
(0 to F, 0 = 16)

size - how many digits to clear
(0 to F, 0 = 16)

CRC Calculator

The crc8 is calculated using the following C routine. You can directly copy and paste this into your own code, either C or Arduino.

source:
http://www.leonardomiliani.com/en/2013/un-semplice-crc8-per-arduino/

```
//CRC-8 - based on the CRC8 formulas by
// Dallas/Maxim
//code released under the terms of the
// GNU GPL 3.0 license

uint8_t crc8_update(uint8_t crc, uint8_t
data) {
  uint8_t updated = crc ^ data;
  for (uint8_t i = 0; i < 8; ++i) {
    if ((updated & 0x80 ) != 0) {
      updated <<= 1;
      updated ^= 0x07;
    }
    else {
      updated <<= 1;
    }
  }
  return updated;
}
```

And the example code below illustrates how to call this routine to compute for crc8 in your led controller driver code. txstring should contain all the characters in the transmit packet, but excluding [STX], [ETX].

Important: crc return value is ORed with 0x80 from its true value. This is necessary to prevent cal_crc8 from generating a value that may return an [STX], [ETX] or 0.

```
uint8_t  cal_crc8(char *txstring){
   uint8_t  crc=0;
   char  *txptr=txstring;
    while(*txptr !=0){

crc=crc8_update(crc,(uint8_t)*txptr);
      txptr++;
    }
   return(crc | 0x80); // see text
}
```

ARDUINO EXAMPLE CODES

Led controller driver routine.

```
// comment out if you do not wish to enable crc checking

#define   CRC_ENABLE

/*
 * Send a packet to Serial Display unit
 * The packet is wrapped with STX and ETX markers as required by the
 * Serial Display Unit
 * txstring - message string to send
 */

 #define  STX 0x02
 #define  ETX 0x03

void  send_to_display(char addr, char *txstring){
  uint8_t  crc=0;
  char  *txptr=txstring;

#ifdef CRC_ENABLE

    //compute for crc8
    crc=crc8_update(crc,addr+0x30); // include device id in the calculation
   while(*txptr !=0){
      //Serial.print(*txptr,HEX);
      crc=crc8_update(crc,(uint8_t)*txptr);
      txptr++;
    }

  crc |= 0x80;
#endif

  Serial.write(STX);
  Serial.write(addr+0x30);
  Serial.print(txstring);

#ifdef CRC_ENABLE
  Serial.write(crc);
#endif

  Serial.write(ETX);

}
```

Led controller initialization. You should call this function within your setup() codes.

```
#define   LEDADDR      5      // led controller device ID. Change according to your own
                // setting


/*
 * Initialize and test Serial Display unit
 */
void  init_display(void){

  // setup serial LED for 16 panels
  send_to_display(LEDADDR,"P0");    // 0=16 panels
  delay(500);
  send_to_display(LEDADDR,"T");     // segment test, turns ON all segment for a second
  delay(1000);

#ifdef CRC_ENABLE
```

```
    // enable crc checking
    send_to_display(LEDADDR,"_CRC");

#endif

}
```

example 1. Clear Display

```
send_to_display(LEDADDR,"C");
```

example 2. Display a “587” (3 digits display) starting in position 5

```
send_to_display(LEDADDR,"D53587");
```

example: Display “3.14156” (6 digits display) starting in position 1

```
send_to_display(LEDADDR,"D163.14156");
```

EXAMPLE PROJECT: BASKETBALL SCORE BOARD

A functional Basketball Scoreboard based or Arduino/Genuino/gizDuino was built as part of test and evaluation of of this device. It uses 16 panels (digits) segmented by software to display team score, team fouls, shotclock and clock timer. Link for the sketch of this project, together with some construction details, will be publish on the product webpage of the Serial LED Display Unit.

Figure 7. Sample Scoreboard project with Serial LED Display system. 16 panels or digits used.

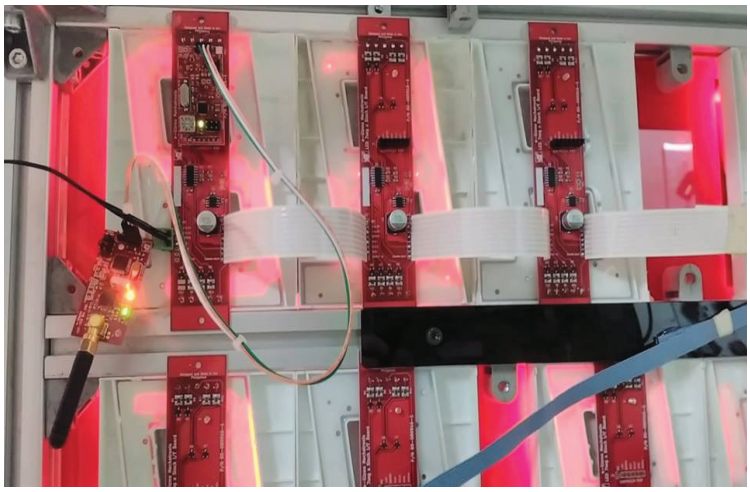


Figure 8. At the back of Scoreboard with 16 panel 7 segment connection with base and expansion panels .It has UHF Data Tranceiver for receiving.

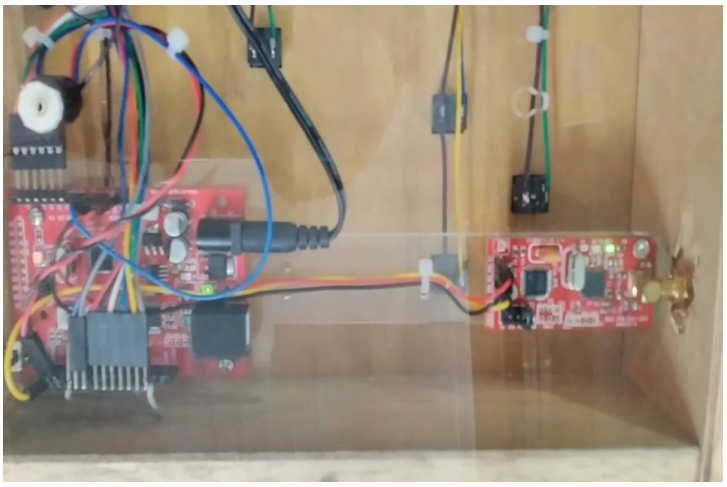


Figure 9. Sample application using gizDuino PLUS with ATMEGA644P and UHF Data transceiver for wireless transmitting.

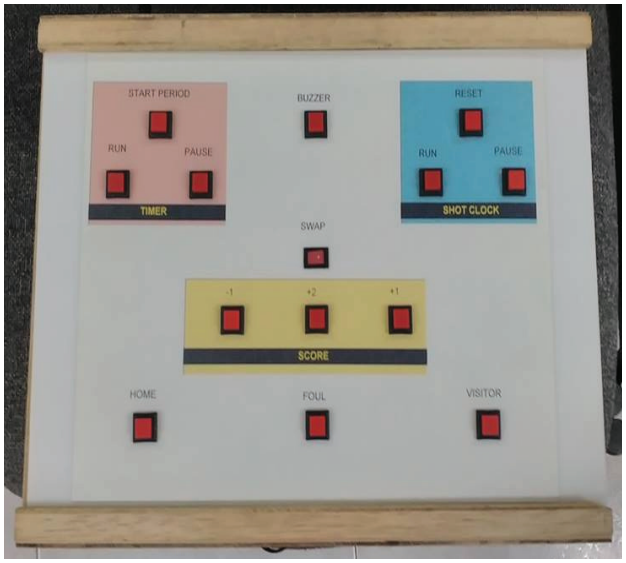


Figure 10. 13 Button panel controls to add score on the display, run/top the timer, team fouls and shotclocks.

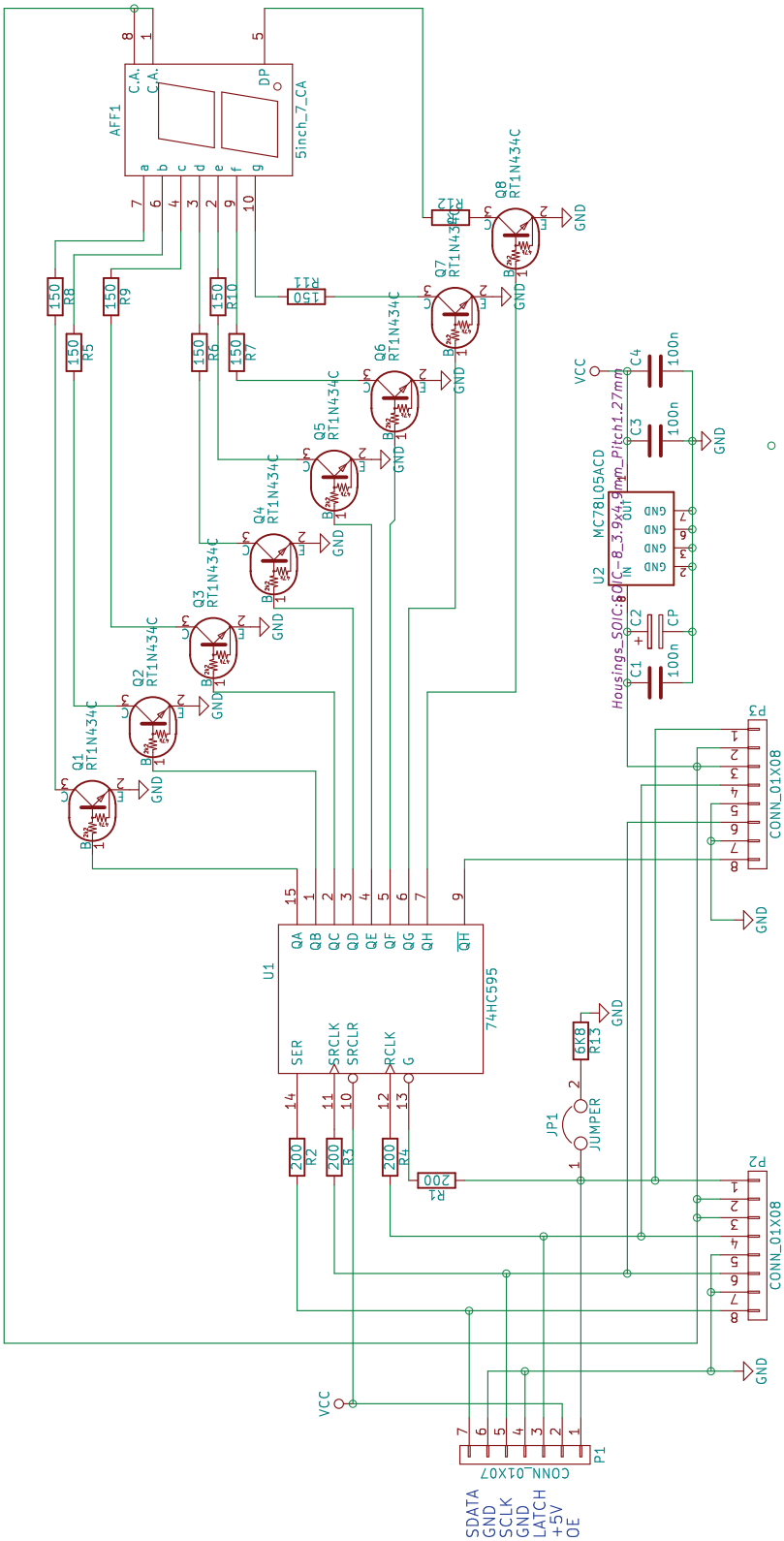


Figure 11. 5-inch Seven Segment Display Panel schematic diagram.

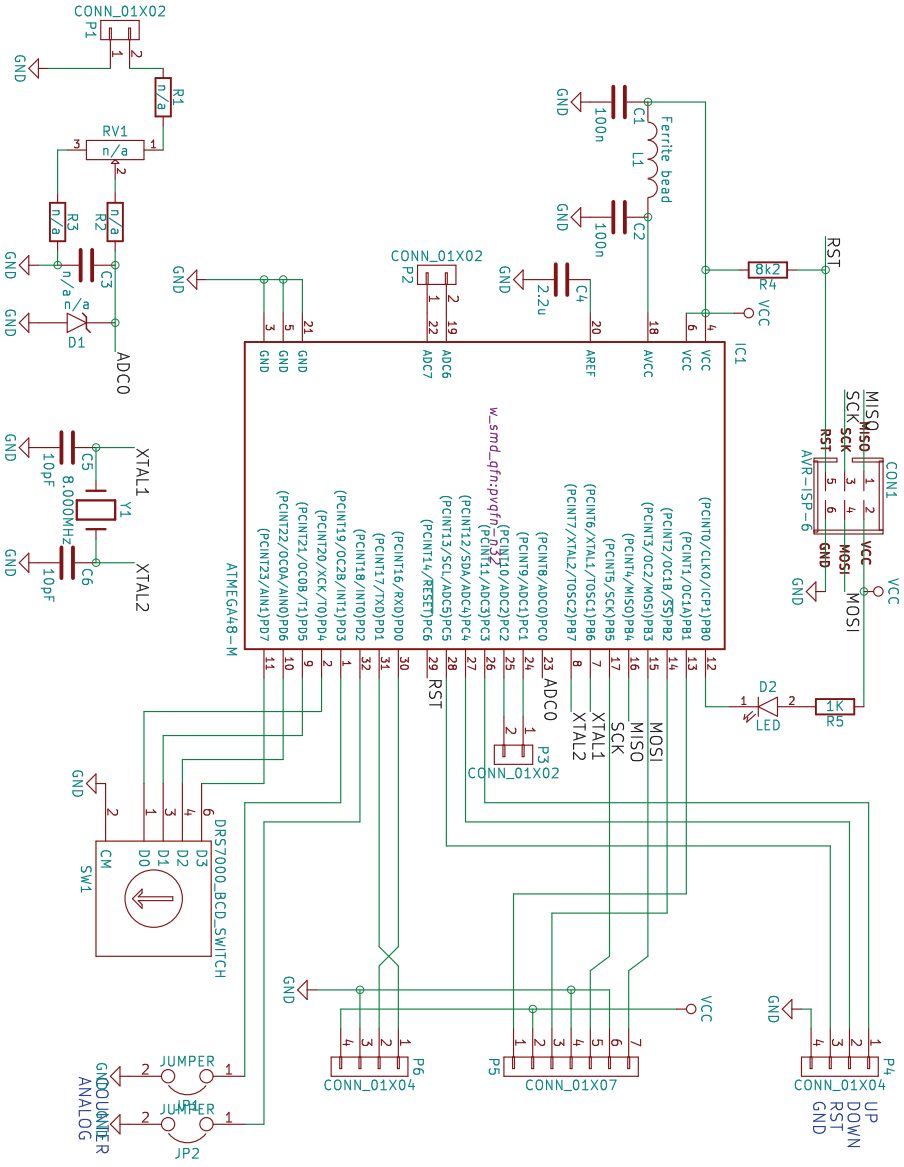


Figure 12. Serial LED Controller Module schematic diagram. One controller module can drive up to 16 display panels.

Contruct this wiring connection then open the 7 segment test.ino.

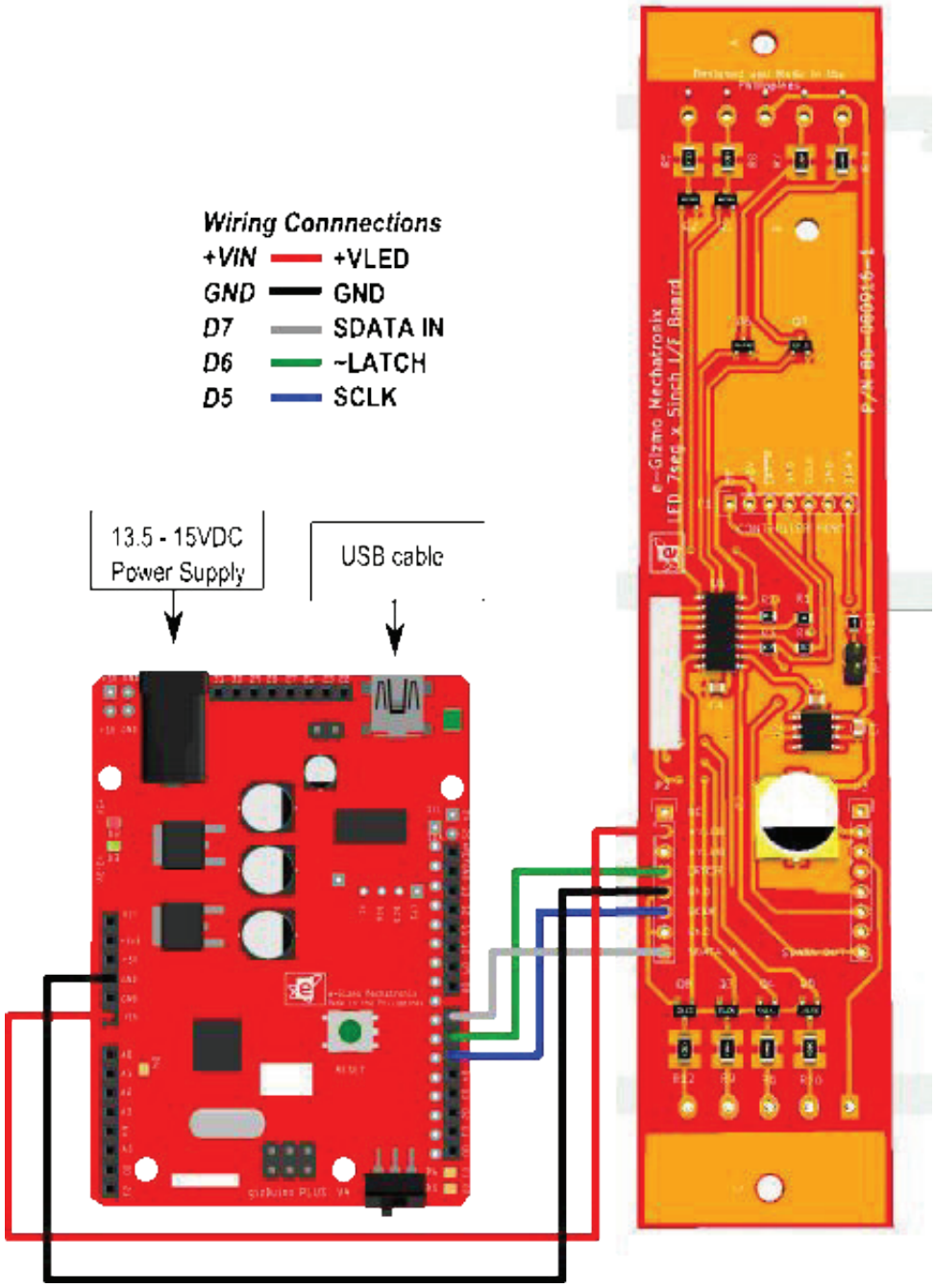


Figure 13. Sample application with gizDuino PLUS ATMEGA644P MCU to Serial 7segment LED 5 inch I/F board.