

e-Gizmo

Temperature Control Unit

Rev. 1r0



Serial LCD II with 2x16 characters LCD display module

LM34(Precision Fahrenheit Temperature Sensor)

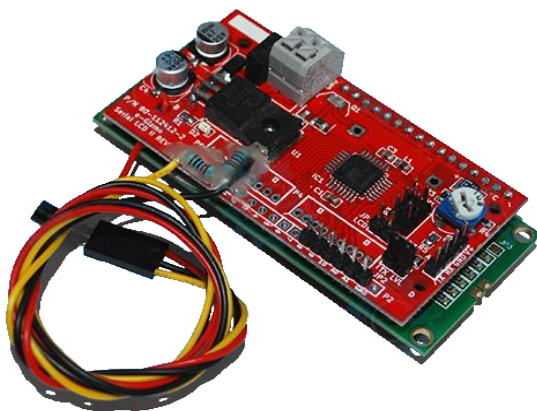


Figure 1. Temperature Controller Unit with LM34 Available.

The e-Gizmo Temperature Control Unit Display is made for easy and fast measurement circuit temperature sensor, whose output voltage is linearly proportional to the Fahrenheit on its LM34 IC sensor then converted/ calibrated directly in degrees celcius to display on the 2 x 16 LCD.

The LM34 device is rated to operate over -50 degrees to 300 degrees Fahrenheit temperature range. The low output impedance, linear output and precise inherent calibration of the LM34 device makes interfacing to readout or control circuitry especially easy.

General Specifications:

Input Supply: 7 to 9 VDC

Maximum Current: 1A

Sensor: LM34 Temperature sensor

Features:

- 2 x16 LCD Display
- Calibrated Directly in Degrees Celcius
- Linear 10.0 mV/degC Scale Factor
- Rated for full -50 deg to 300 deg F range
- UART control (TTL Level)
- Suitable for remote applicationsa and Battery management
- Arduino Compatible

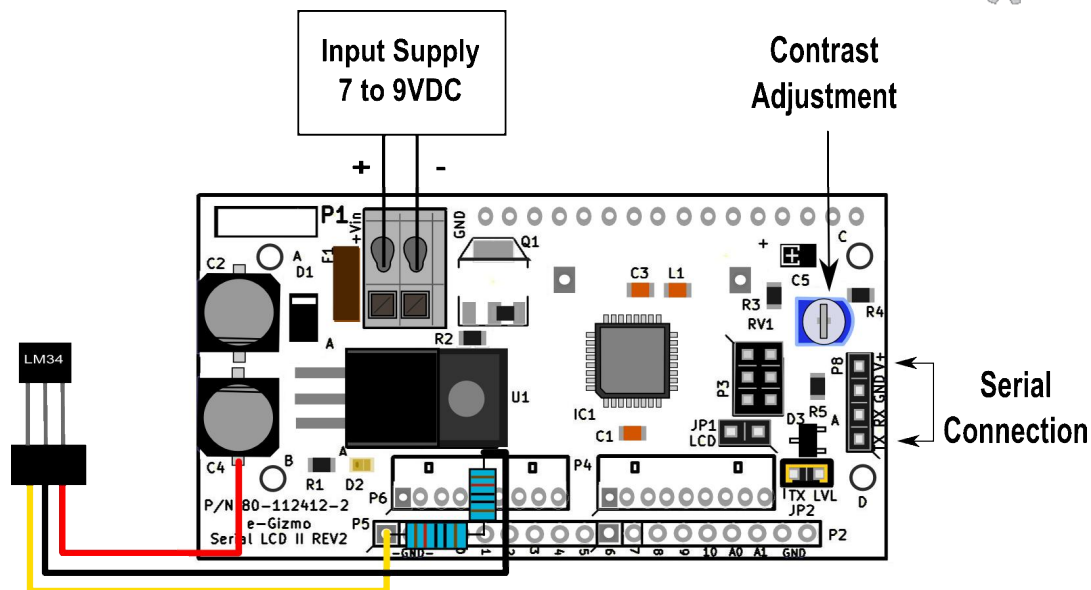


Figure 2. Major Presentation of Temperature Controller Display Unit

LP Package
3-Pin TO-92
(Bottom View)

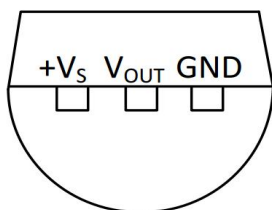


Figure 3. Pin configuration and Function of LM34.

TABLE 1. PIN FUNCTIONS

Name	Type	Description
+VS	POWER	Positive power supply pin
Vout	OUTPUT	Temperature Sensor Analog Output
GND	GROUND	Device ground pin, connect to power supply negative terminal

TABLE 2. SERIAL CONNECTIONS

Name	Type	Description
+V	POWER	+5V supply
GND	GROUND	Device ground pin, connect to power supply negative terminal
RX	Received	Received commands from MCU/Terminal
TX	Transmit	Transmitting data from Serial LCD II to MCU/Terminal

TABLE 3. INPUT SUPPLY CONNECTIONS

Name	Type	Description
+Vin	POWER	+5V supply, positive terminal
GND	GROUND	Device Ground pin, negative terminal

1. PREPARATION FOR USE

The TCU (Temperature Control Unit) also borrowed the PCB of an existing kit of Serial LCD II board with LM34 Temperature sensor are already installed.

Using USB to TTL converter is the easiest way to communicate from PC to TCU board.

1.1 Connect the USB to TTL converter

1.2 Power up the Serial LCD II, Input Supply is 7 to 9VDC.

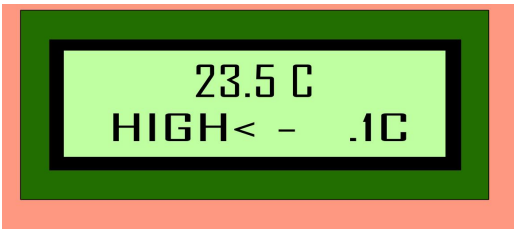


Figure 4. Start Up (not calibrated).

2. FOR CALIBRATION

Setting Up

1. Install the terminal on your PC.
(Terminal v 1.9b by Bry@++)
2. Connect the USB to TTL converter to TCU.
3. Make sure you powered the Serial LCD II.

To operate the Temperature Control Unit Display, you need to calibrate your unit.

USB to TTL ---> Serial LCD II wiring connection

+5V	----> +V
GND	----> GND
TXD	----> RXD
RXD	----> TXD

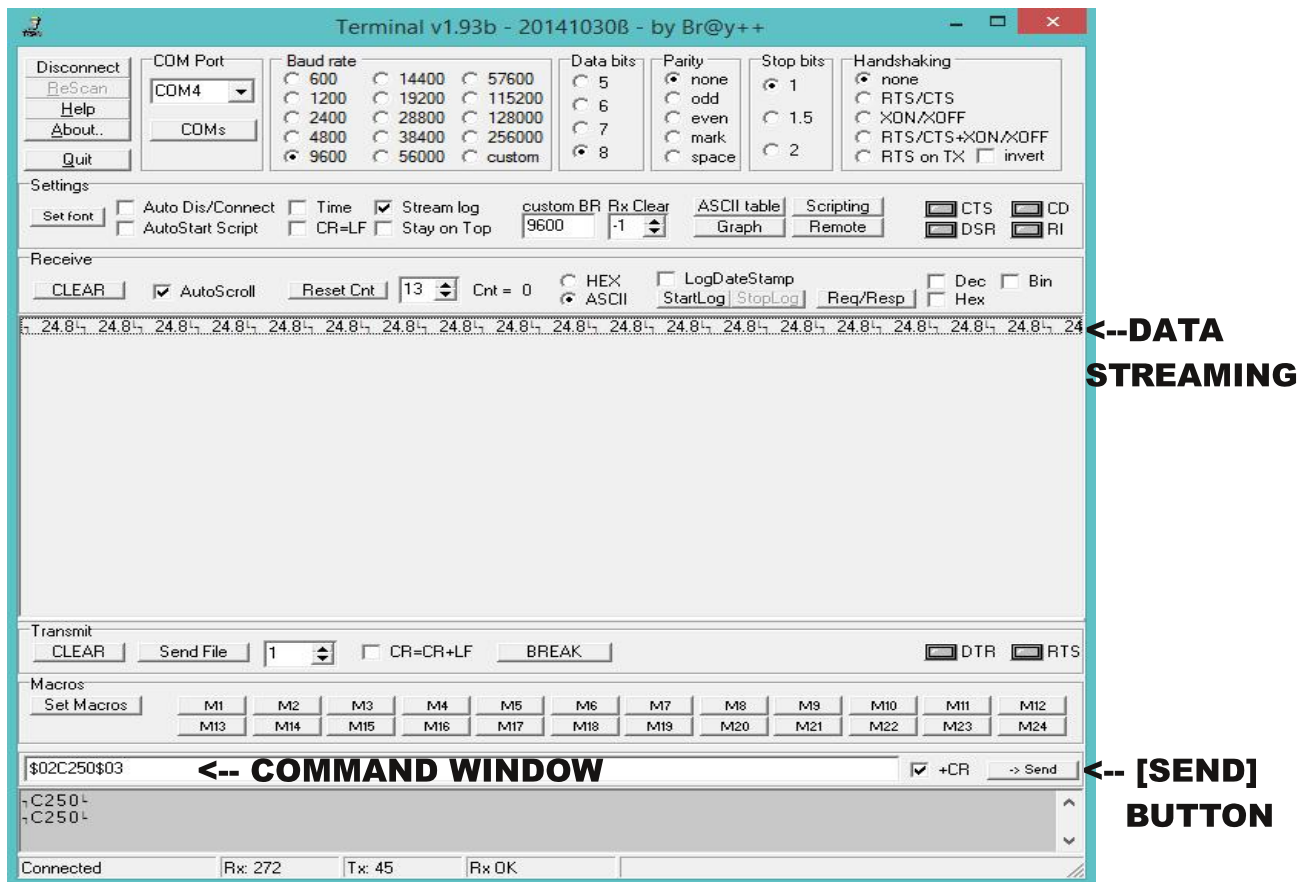
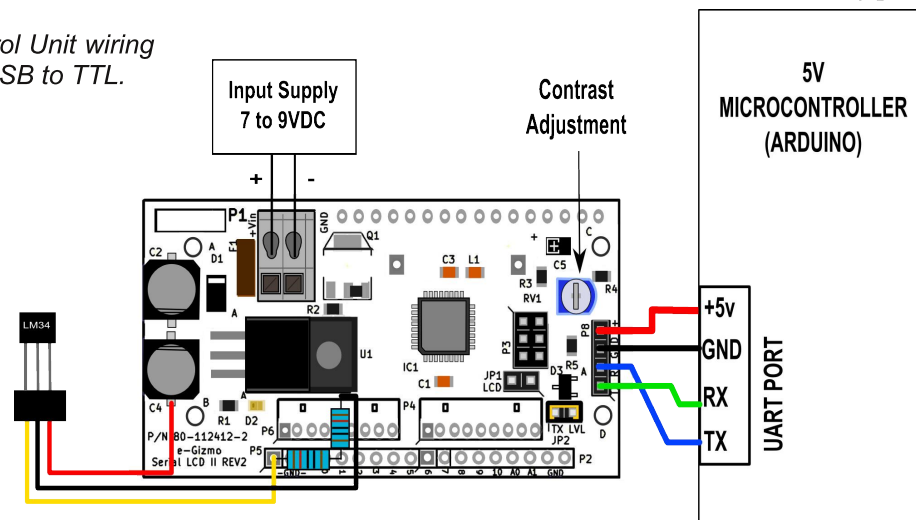


Figure 5. Temp. Control Unit wiring connection to MCU/ USB to TTL.



COMMUNICATIONS MANUAL

Baud Rate: 9600
Data: 8 Bit
Parity: none
Handshake: none

Summary of Functions

H - Set High trigger for comparator
L - Set Low trigger for comparator
C - Calibrate to this temperature
S* - Save current settings (to NVRAM)
T - test

Important:

Communications Format

Every packet of data transmission are wrapped inside an [STX] and [ETX] marker.

[STX] – Start of transmission marker, ASCII value = 0x02
[ETX] – End of transmission marker, ASCII value = 0x03

The first character after the [STX] marker 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 signaled by the [ETX] marker.

[STX] and [ETX] are data packet markers and should not be transmitted as a literal string. They should be sent in their ASCII representation. The

correct way of transmitting the [STX] and [ETX] markers are as shown in the following example:

Example 1: Test

Transmission Format: Format: [STX]T[ETX]

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

Symbol	STX	T	ETX
Hex	0x02	0x54	0x03

Visual Basic:

Correct:

' correct way to send [STX] & ETX marker
`Serial1.print(chr(2)+"T"+chr(3))`

Wrong:

`Serial1.print("[STX]T[ETX]")` 'WRONG!

Arduino:

Correct:

`Serial.write(0x02); //correct way to send [STX]`

`Serial.print("T");
Serial.write(0x03); // [ETX] marker`

Wrong:

`Serial.print("[STX]T[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("\002T\003");    //"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

Important: All settings made through the following functions are temporary and will return to its old saved value once the TCU is power cycled. If you want to keep the current settings as your new instrument saved value, you should perform the Save settings (no. 4) function.

1. **H** - Set HIGH trigger for comparator

Format:[STX]**H**nnn[ETX]
Reply: [STX]OK[ETX]

where: nnn - 3 digit set value, 0-999
Note: Do not include a decimal point in the entry.

Example:

With the display unit decimal point set for “00.0” display
[STX]H123[ETX] will set the high trigger to 12.3

2. **L** - Set LOW trigger for comparator

Format:[STX]**L**nnn[ETX]
Reply: [STX]OK[ETX]

where: nnn - 3 digit set value, 0-999
Note: Do not include a decimal point in the entry.

3. **C** - Calibrate to “this” temperature”

To calibrate the Temperature Control unit,

Format:[STX]**C**nnn[ETX]
Reply: [STX]OK[ETX]

where nnn = 3-digit test sample temperature
Note: Do not include a decimal point in the entry.

4. **S*** - Save current settings

All new settings entered via the UART functions are temporary in nature and will be lost once the power is cycled. This will allow the users to programmatically tweak the behavior of the TCU without permanently losing the saved settings. If you wish to reconfigure your TCU to power up with this new setting, the Save S* request must be sent.

Format:[STX]**S***[ETX]
Reply: [STX]OK[ETX]

Previously saved settings will be lost and replaced by the new settings.

5. **T** - Test

Used primarily to test the UART port.

Format:[STX]**T**[ETX]
Reply: [STX]OK[ETX]



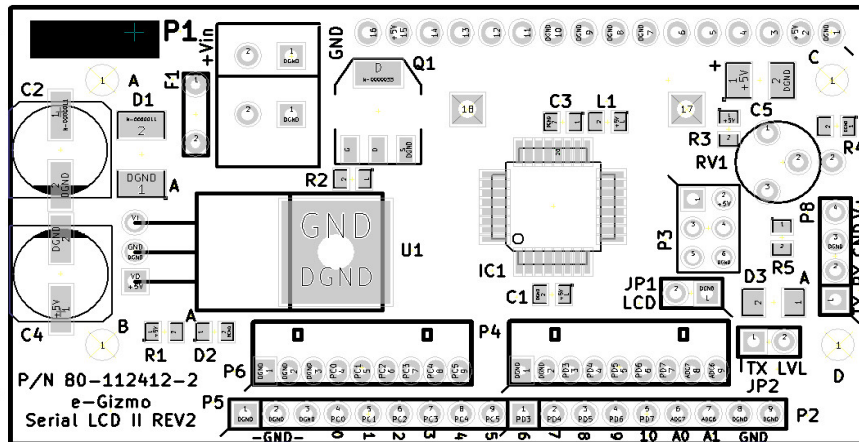


Figure 6. Silkscreen Guide

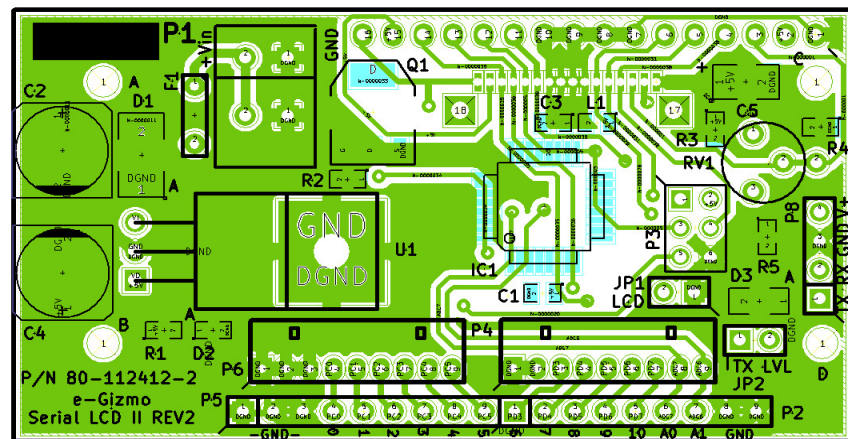


Figure 7. Bottom PCB Layer

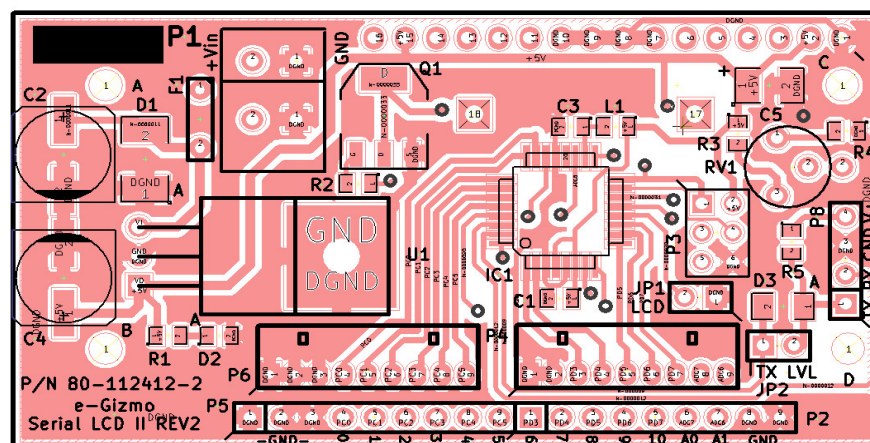


Figure 8. Top PCB Layer

SCHEMATIC DIAGRAM

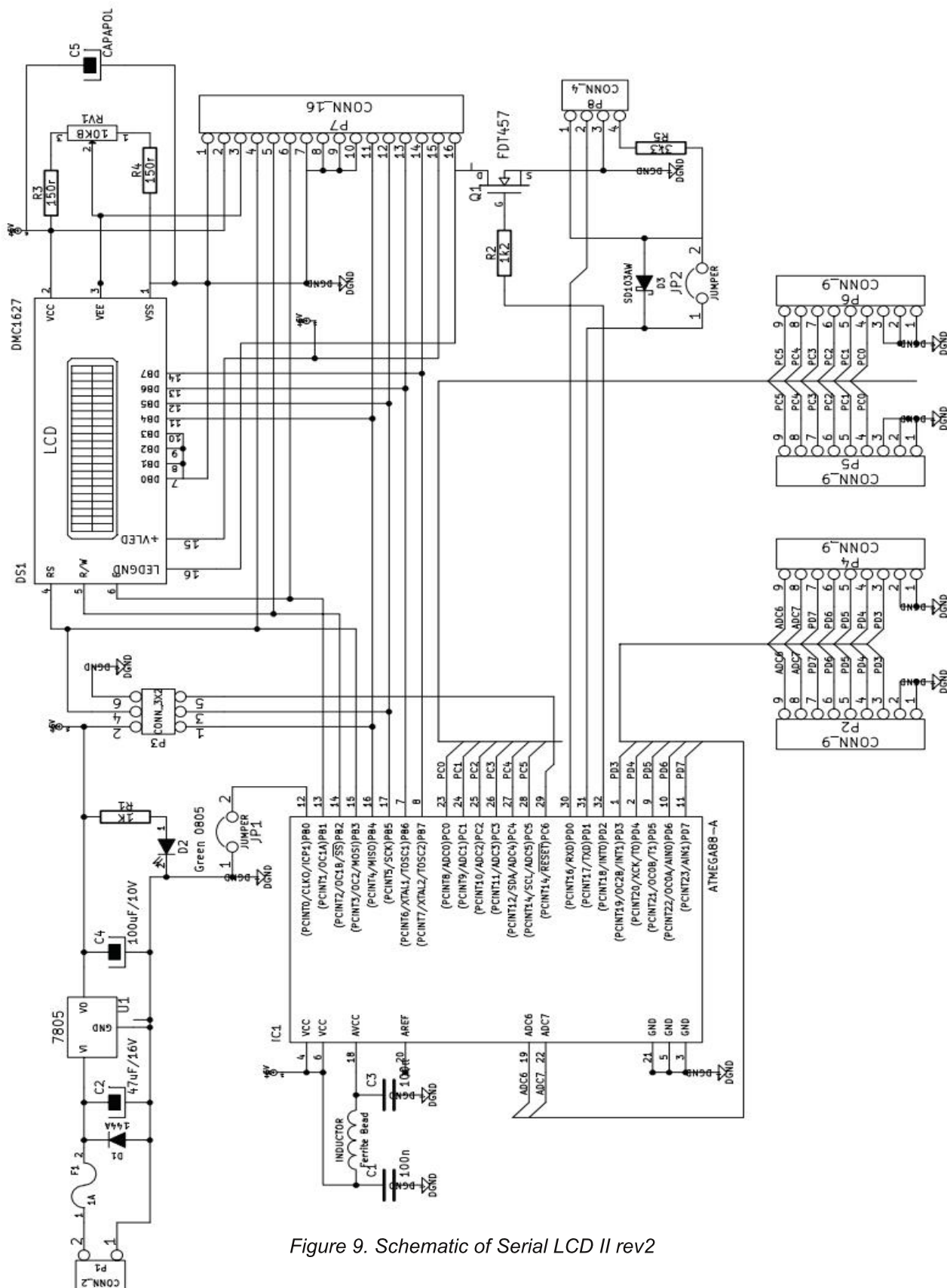


Figure 9. Schematic of Serial LCD II rev2

SAMPLE DISPLAY APPLICATIONS

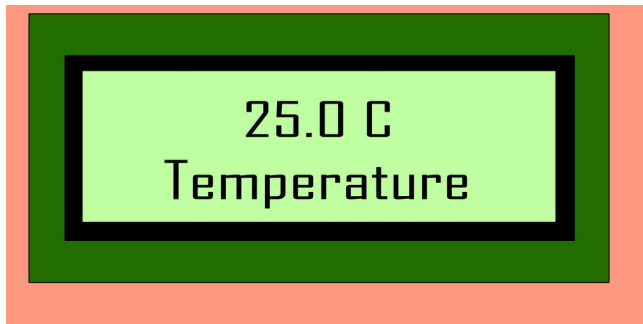


Figure 10. StartUp Display

Calibration:

if your test Temperature is 25.0 C.

\$02C250\$03

**One decimal point*

\$02L230\$03

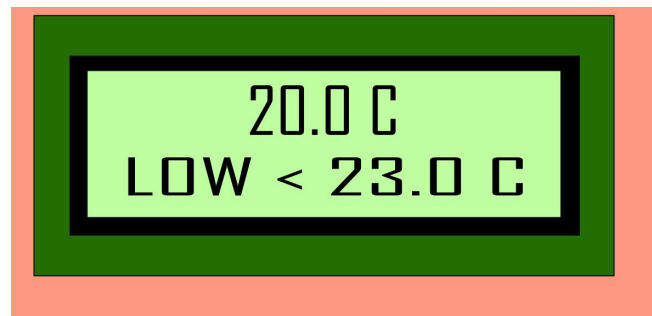


Figure 11. SettingUp LOW trigger comparator

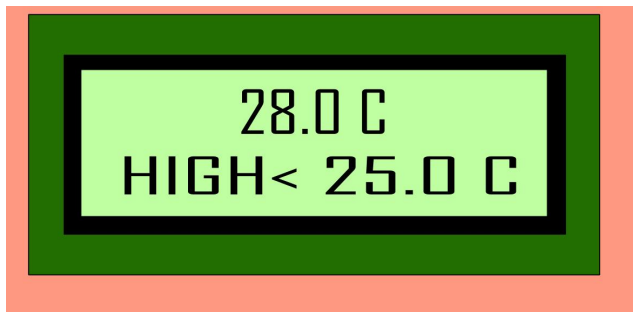
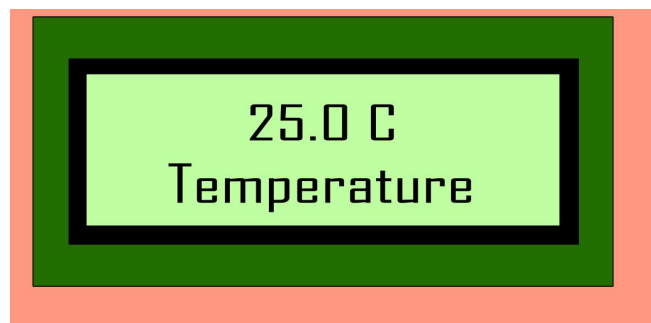


Figure 12. SettingUp HIGH trigger comparator

\$02H250\$03

\$02S*\$03
Save Current settings



SAMPLE APPLICATIONS WITH GIZDUINO + 644P

Wiring Connections:

Temperature Control Display Unit ----> *gizDuino +*

RX ----> TX0(pin1)
TX ----> RX0(pin0)
GND ----> GND(Ground)
+V ----> +5V

Wiring Connections:

Temp. control Display Unit ----> *LEDs Indicators*

pin4(D4) ----> "Low" Comparator
pin5(D5) ----> "High" Comparator
GND ----> GND(Ground)

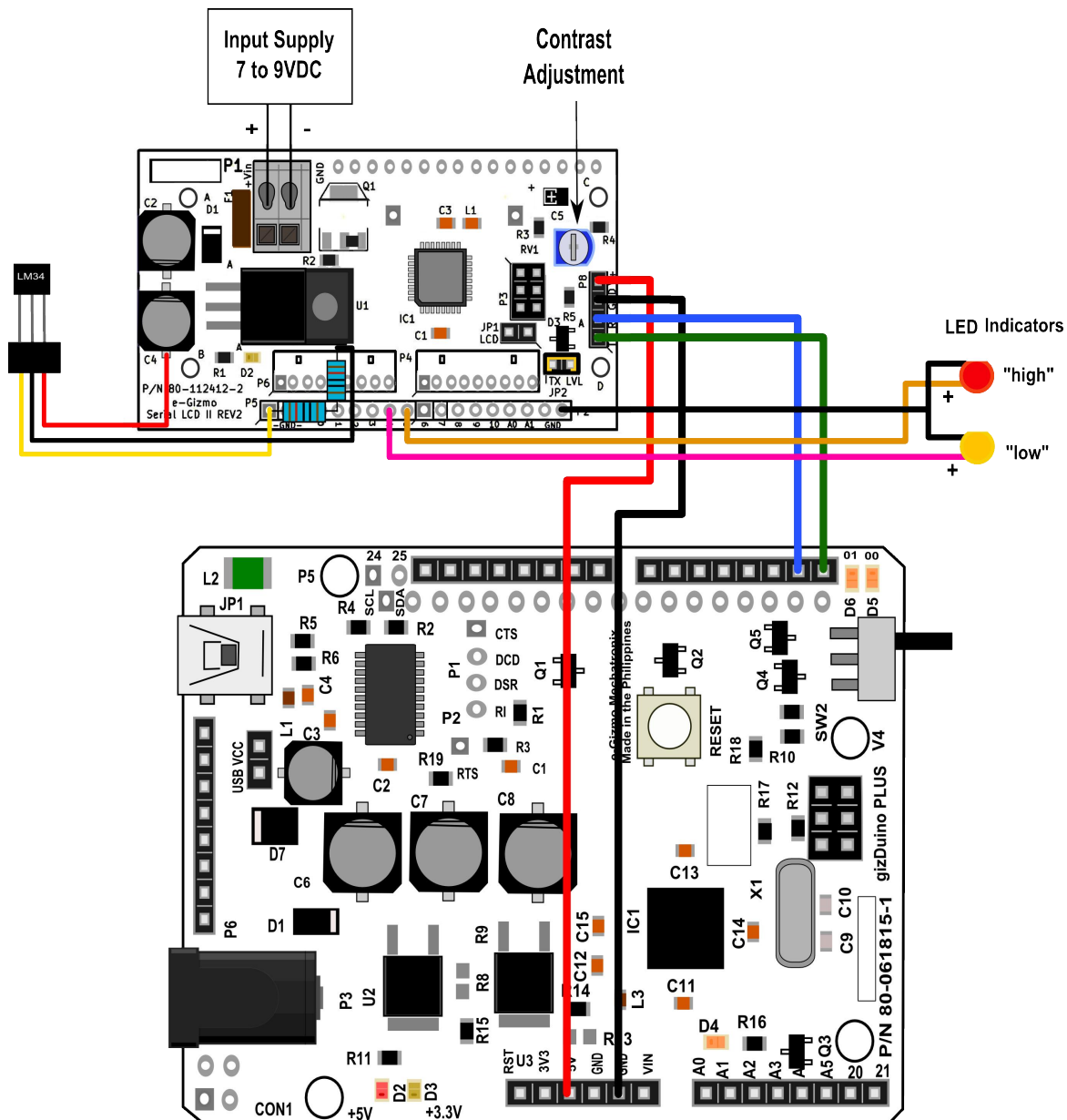


Figure 13. Sample Application of Temperature Control Display Unit with gizDuino + w/ ATmega644P