8 Character 7-Seg Display - HT16k33 - Trēo™ Module

Module Features

- Holtek HT16k33
- 8 Digit Numerical Display
- 16-Step Dimming
- RoHS Compliant
- Software Library
- NightShade Trēo™ Compatible
- Breakout Headers

Applications

- Laboratory Measurements
- Industrial Control
- Displays

Trēo™ Compatibility

Electrical

Communication	I2C
Max Current, 3.3V	1mA
Max Current, 5V	250mA

Mechanical

- 115mm x 25mm Outline
- 110mm x 20mm Hole Pattern
- M2.5 Mounting Holes

Description

The HT16k33 Trēo™ Module is a 8 Character 7-Seg Display module that features Holtek's HT16k33 8 Character 7-Seg Display. It receives instructions from the host and handles the switching and multiplexing of the display. This module is a part of the NightShade Treo system, patent pending.

Table of Contents

1 Summary 2
2 What is Trēo™? 2
3 Electrical Characteristics 2
4 Electrical Schematic 3
5 Mechanical Outline 4
6 Example Arduino Program 5
7 Library Overview (C++ & Python) 6

1 Summary

The HT16K33 7-segment display is initialized with the begin() method. Then a local display buffer is written using the printNumber(), setDigit(), setDecimal(), and clearDisplay() methods. Finally, the local buffer is written to the display with the writeDisplay() method and the characters will be displayed on the 7-segment LED display.

2 What is Trēo™?

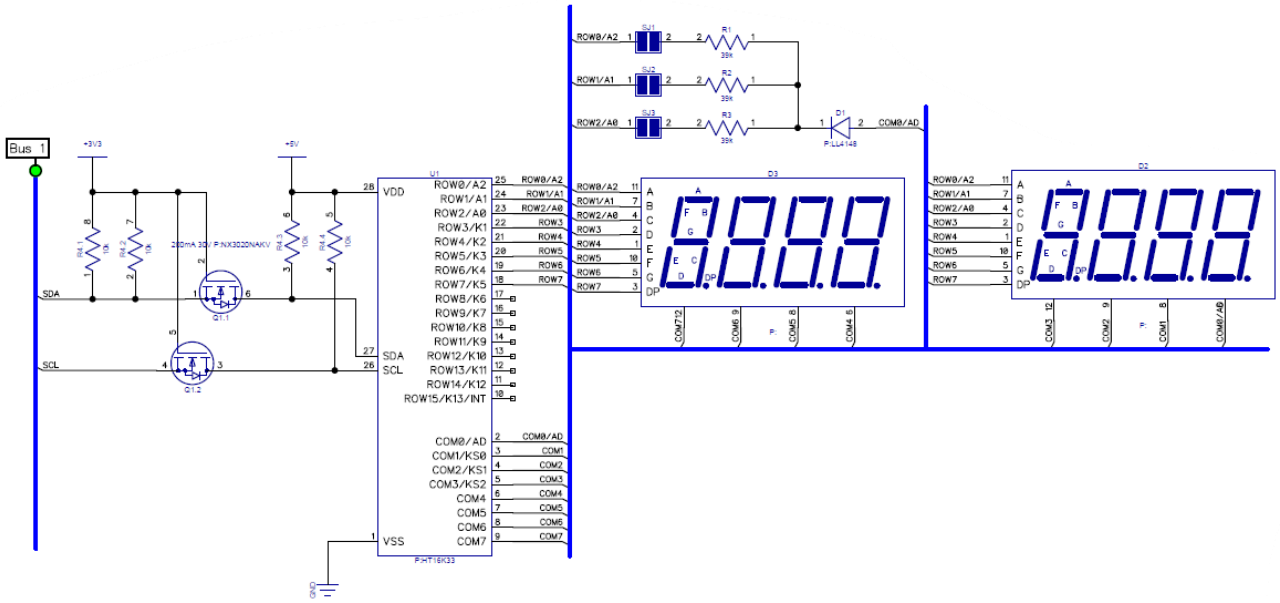
NightShade Trēo is a system of electronic modules that have standardized mechanical, electrical, and software interfaces. It provides you with a way to quickly develop electronic systems around microprocessor development boards. The grid attachment system, common connector/cabling, and extensive cross-platform software library allow you more time to focus on your application. Trēo is supported with detailed documentation and CAD models for each device.

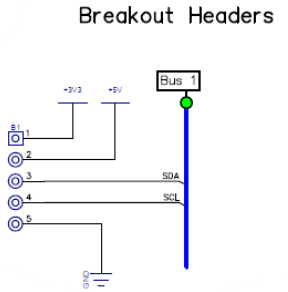
Learn more about Trēo here.

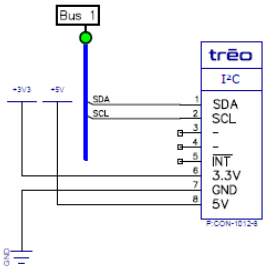
3 Electrical Characteristics

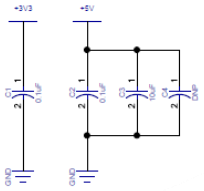
	Minimum	Nominal	Maximum
Voltages			
$V_{i/o}$ (SDA, SCL, INT)	-0.3V	-	3.6V
$V_{3.3V}$	3.1V	3.3V	3.5V
V_{5V}	4.8V	5.0V	5.2V
I2C Slave Address			
SJ1-SJ3 Open (Default)		0x70	
Alt. Address (Soldered=1)		B 1 1 1 0 [SJ3] [SJ2] [SJ1]	

4 Electrical Schematic

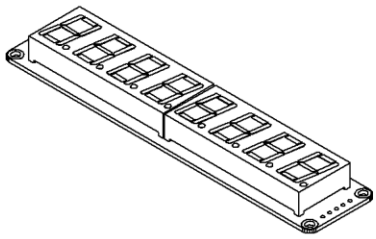


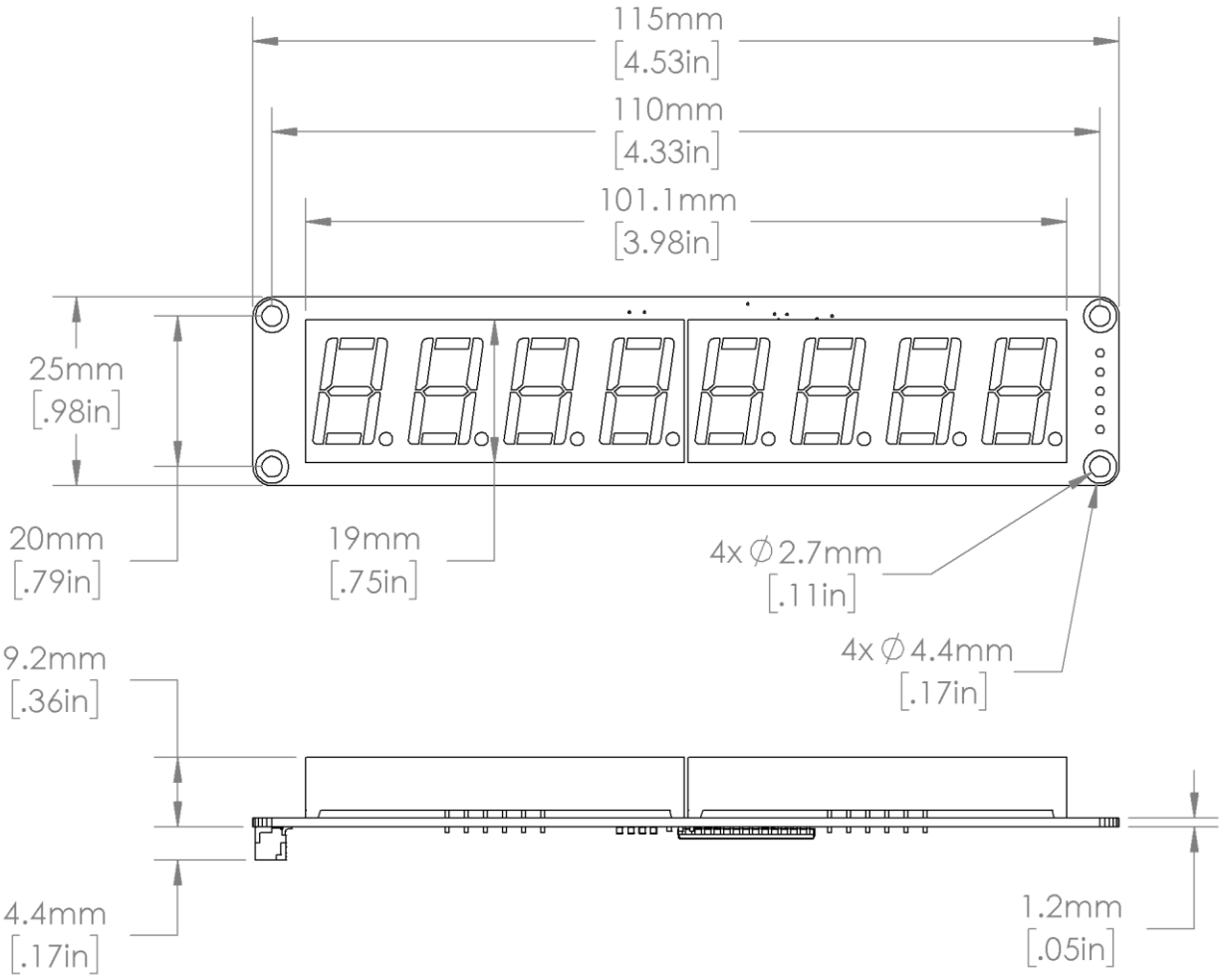






5 Mechanical Outline





6 Example Arduino Program

```
/*********************************************************
  HT16K33_7SegmentDisplay - NightShade_Treo by NightShade Electronics

  This sketch demonstrates the functionality of the
  NightShade Trēo HT16K33 4 or 8 character 7-segment
  display module. (NSE-1157-1/2) It prints the program time
  to the display in seconds.

  Created by Aaron D. Liebold
  on February 15, 2021

  Links:
  NightShade Trēo System: https://nightshade.net/treo
  Product Page (8 Character): https://nightshade.net/product/treo-8-character-7-seg-
display-ht16k33/
  Product Page (4 Character): https://nightshade.net/product/treo-4-character-7-seg-
display-ht16k33/

  Distributed under the MIT license
  Copyright (C) 2021 NightShade Electronics
  https://opensource.org/licenses/MIT
*********************************************************/

// Include NightShade Treo Library
#include <NightShade_Treo.h>

// Declare Objects
NightShade_Treo_HT16K33 display(1);

void setup() {
  display.begin();
}

void loop() {
  display.clearDigits();
  display.printNumber((float) millis() / 1000, 2);
  display.writeDisplay();
}
```

7 Library Overview (C++ & Python)

C++ Class

NightShade_Treo_ HT16K33 <classObject>();

Python Module

<classObject> = NightShade_Treo. HT16K33()

7.1 Constructors

NightShade_Treo_HT16K33(int port, uint8_t slaveAddress, uint32_t clockSpeed)

Creates a HT16k33 object.

Arguments:

port	Integer of the I2C port used (e.g. 0 = "/dev/i2c_0")
slaveAddress	7-bit slave address
clockSpeed	Desired clock speed for the bus

Returns:

Nothing

NightShade_Treo_ HT16K33(int port)

Creates a HT16k33 object assuming the default slave address and clock speed.

Arguments:

port	Integer of the I2C port used. (e.g. 0 = "/dev/i2c_0")

Returns:

Nothing

7.2 Methods

begin()

Initializes the HT16K33.

Arguments:

None

Returns:

Error	0 = Success

sleep(int enable)

Disables the HT16K33's internal oscillator.

Arguments:

enable	true/false

Returns:

Error	0 = Success

printNumber(double value, int decimalPlaces, int offset)

Prints the value of a number to the display buffer with the designated number of decimal places. The number be shifted on the display with the *offset* value.

Arguments:

value	numerical value (integer or fractional)
decimalPlaces	number of fractional digits to be displayed
offset	number of digits to shift the display right

Returns:

Nothing

printNumber(double value, int decimalPlaces)

Prints the value of a number to the display buffer with the designated number of decimal places.

Arguments:

value	numerical value (integer or fractional)
decimalPlaces	number of fractional digits to be displayed

Returns:

Nothing

printNumber(double value)

Prints the value of a number to the display buffer.

Arguments:

value	numerical value (integer or fractional)

Returns:

Nothing

setDigit(int digit, int8_t value)

Writes a value to a single digit in the display buffer. Bits 0-3 hold the digits value of 0-9. Setting any bit (B4-B5) will enable the decimal point of the selected digit. (e.g. Add 16 to the digit's value to set B4.)

Arguments:

digit	0-7
value	0-9 (Adding 16 to 0-9 enables DP)

Returns:

Nothing

setDecimal(int digit, int enable)

Sets a the DP state at the selected digit in the display buffer.

Arguments:

digit	0-7
enable	true/false

Returns:

Nothing

clearDigits()

Clears the display buffer.

Arguments:

None

Returns:

Nothing

writeDisplay()

Writes the display buffer to the 7-segment display.

Arguments:

None

Returns:

Error	0 = Success

setOnBlinking(int screenEnable, int blinkingValue)

Enables the display output and sets a the blink rate of the display.

Arguments:

screenEnable	true/false
blinkingValue	0: No Blinking (Constant ON) 1: 2Hz 2: 1Hz 0.5Hz

Returns:

Error	0 = Success

setBrightness(uint8_t brightness)

Sets the brightness of the display.

Arguments:

brightness	0-15

Returns:

Error	0 = Success