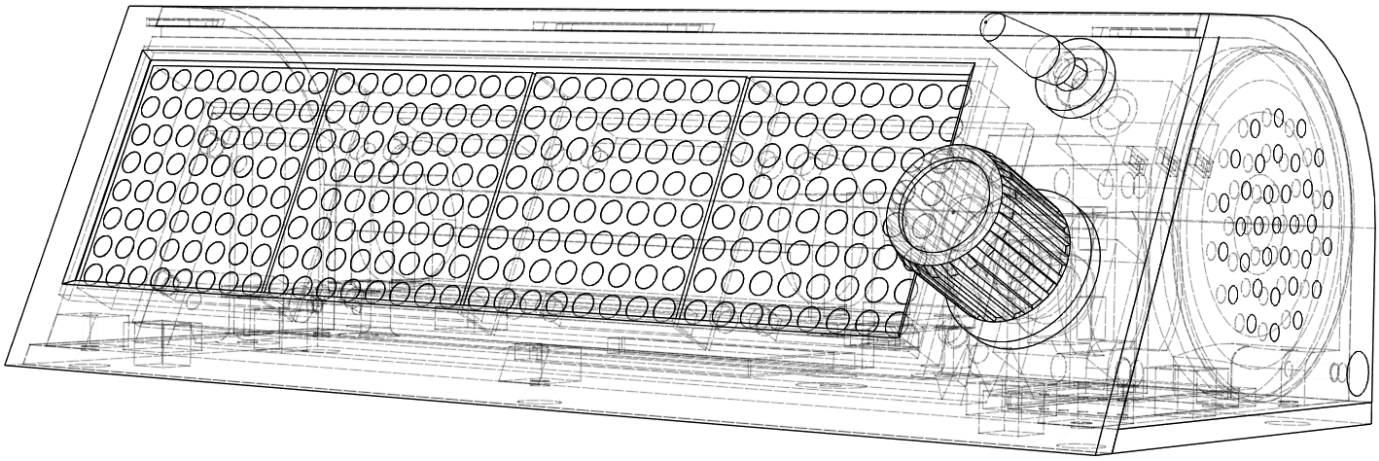
Clock

Build Manual



This document contains information how to build a clock from basic building modules and using a 3D printer to create the case. This clock has several functions, it is meant as a 'kitchen timer' but includes also a stopwatch and a normal alarm time. The software is included, to be changed to your liking. The user control uses a single rotary/push button. The display is a large dot matrix of four 8x8 pixel modules.

Copyright © 2022, Henk Beekhof All rights reserved.

Table of contents

1 THE SETUP / USED ITEMS 3
1.1 SCHEMATIC 3
1.2 THE HW BOARDS AND ELECTRICAL COMPONENTS 4
1.3 THE MECHANICAL COMPONENTS 4

2 WHERE TO FIND WHAT 5

3 BUILD INSTRUCTIONS FOR THE HARDWARE 6
3.1 STEP 1: THE CASE 6
3.2 STEP 2: SCHEMATIC, MOUNT ELECTRICAL COMPONENTS 7
3.3 STEP 3: PROGRAMMING AND TESTING 11

4 ARDUINO IDE 12
4.1 INSTALLATION 12
4.2 LOCATION OF THE CODE + WAY OF WORKING 12
4.3 SETUP BOARD AND PORT TO COMPILE AND UPLOAD SOFTWARE 12

5 THE SOFTWARE 14
5.1 LIBRARY DEPENDENCIES 14
5.2 OPERATION 14

6 OPERATING MANUAL 15

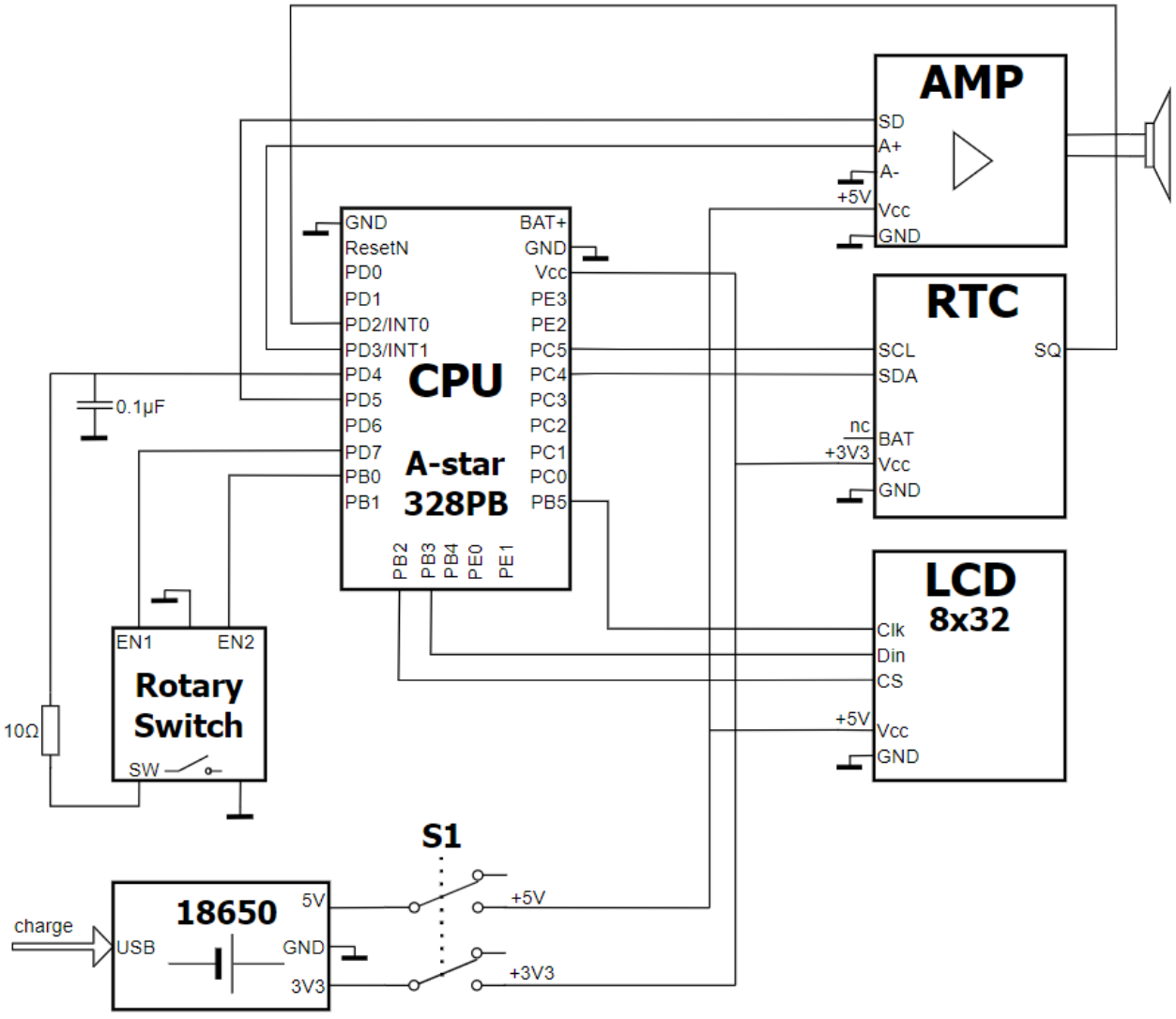
7 DETAILED INFO - HW BOARDS 16
7.1 A-STAR 328PB MICRO AND USB AVR PROGRAMMER V2.1 16
7.2 FC16 MATRIX LCD (MAX7219) 18
7.3 RTC/EPROM MODULE DS3231 / AT24C32 20
7.4 ROTARY ENCODER + SWITCH 20

 Copyright © 2022, Henk Beekhof All rights reserved.

1 The setup / used items

This chapter lists the different items required to build the clock.

1.1 Schematic



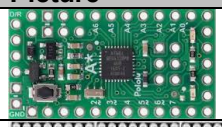
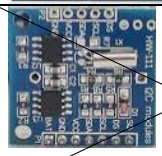
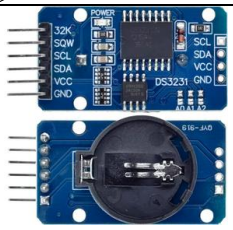
Note: The rotary drawing corresponds with a backside view

The hardware setup is simple. The microcontroller acts on a 1Hz pulse from the RTC, this is the system heartbeat. The Rotary/Switch and LDC form the user interface for input and output. An audio output is implemented by a standard Class D amplifier and a small speaker. The system is powered by a single battery which is mounted on a battery shield which converts the battery voltage to a stable 3.3V and 5.0V supply. This shield contains a charge circuit where the battery is charged from a micro-USB input. The only glue component is an RC to debounce the switch function of the rotary knob.

Copyright © 2022, Henk Beekhof All rights reserved.

1.2 The HW boards and electrical components

The list of used components:

Item	Description	Picture
CPU	A-Star 328PB Micro 3.3V 8MHz Pololu	
LCD	MAX7219 FC-16 64x8 Dot Matrix Display	
~~RTC~~	~~DS1307 / 24C32 - RTC / EEPROM module~~ Battery CR2032 Based on experience and testing I don't recommend to use this module. The crystal's internal load does not match the RTC chip requirements, resulting in an inaccuracy of several seconds per day. For a clock this module is useless.	
RTC	DS3231 RTC / EEPROM module Battery CR2032 **NOTE**: When the battery is used then removing resistor U4, (at the right side, above the text SCL) OR the diode at the left side of this resistor must be removed. The circuit is too much simplified which makes this action mandatory. If kept and Vcc-Vdiodedrop gets higher than Vbat: The battery will be charged. This is absolutely dangerous for lithium batteries.	
AMP	PAM8302 2.5W Class D amplifier module Adafruit	
Speaker	Thin plastic speaker 8 ohm, 0.25W, 39mm, ADA-1891 Adafruit	
18650	18650 battery shield 18650 battery	
Rotary	Rotary encoder (3 pins) + push switch (2 pins) Debounce RC: C = 0.1 µF, R = 10 Ω	
Switch S1	Toggle switch 2-p on/off with solder lugs	

Indication of costs: Some of the boards can be ordered in a set of 5 or 10 pieces. Using different suppliers the cost is around 25 Euros to 30 Euros for the above list of items. Note: Pay attention to where you buy the hardware, you find easily a factor 5 to 10 higher prices from sites which 'only push your order forward, while making money'.

1.3 The mechanical components

Item	Description
Magnet	4x magnet 5x5x5 [mm] Optional
Screw	8x M3x5 black oxide steel philips cross recessed pan head self-tapping screw
Screw	7x M2x8 stainless steel flat head philips self-tapping (Case)
Bolt + Nut	M3 x 23 (to mount the RTC board)
Case bottom	3d printed, file: Bottom_plate_v4.stl
Case front	3d printed, file: Front_v34.stl
Case cover	3d printed, file: Cover_v28.stl
Case knob	3d printed, file: Rotary_knob_v3.stl
Screen	Plexiglass plate: 132 x 35.2 x 2 [mm]

Copyright © 2022, Henk Beekhof All rights reserved.

2 Where to find what

On the website a zip file can be downloaded. The content of the zip file is described on this page.

- Clock_v30.doc	This document
- Clock_timer_schematic.png	Schematic
- Dir: *Clock_Case_STL_files*	3D print files for the 4 case items
Bottom_plate_v4.stl	
Cover_v28.stl	
Front_v34.stl	
Rotary_knob_v3.stl	
- Dir: *Operating_Manual*	Manual of the clock / Timer / Stopwatch / … functions
Operating_Manual.jpg	
- Dir: *Software*	The Arduino project directory / SW source code
Clock_v20	Directory containing the sketch
project_libraries	Directory with used libraries as reference
Clock_v30.ino	Arduino IDE project file
defs.h	Definitions of pins, modes, etc.
font3x5.h	Definition of font
font4x7digits.h	Definition of font of numbers 0 to 9
license.txt	MIT license (free usage)
pitches.h	Definition of sounds
Rotary_wemos.cpp + .h	Rotary class
Time.cpp + .h	Time_Interface class

Copyright © 2022, Henk Beekhof All rights reserved.

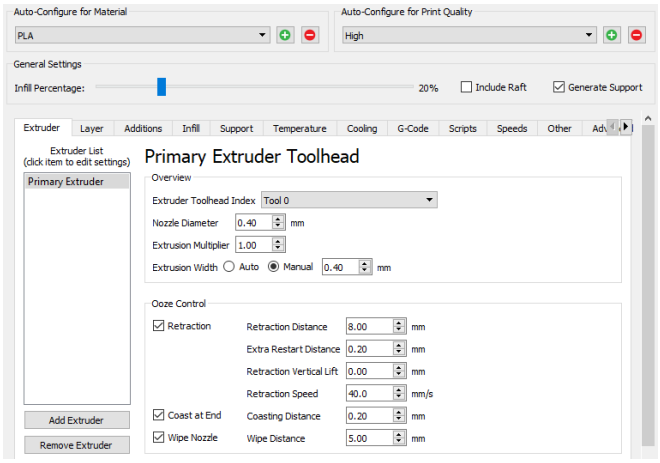
3 Build instructions for the hardware

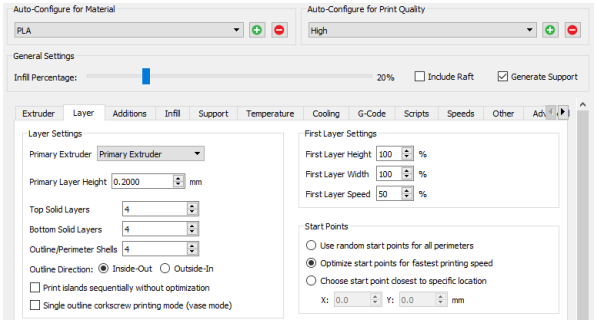
This is a guideline for building the hardware, with the intention to highlight items not seen in the schematic.

3.1 Step 1: The case

Print the four parts of the case or order them by a print service.
The STL files are provided, slice them with your preferred software to match your 3D printer.
I have used a Creality CR10S printer and Simplify 3D as slicer software.

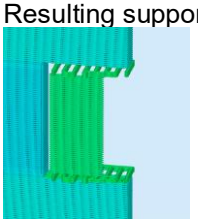
All items are printed with a 0.4mm nozzle, using 0.2mm layer height, the basic settings are shown here:

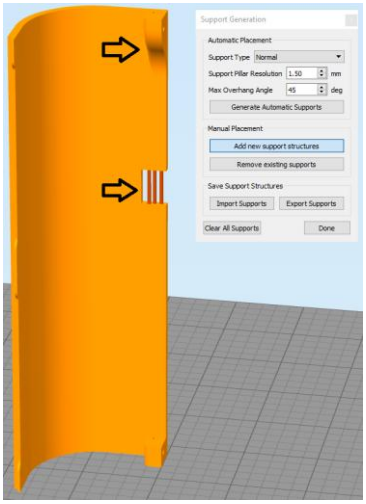




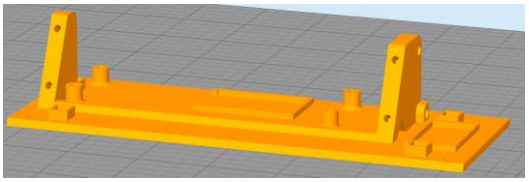
Cover:
The cover is intended to be printed in the orientation shown in the picture. See the upper black arrow, that piece should be at the top side, it gradually widens to hold the mounting screws. This eliminates the need for support material at this location.

The cover needs support material for the USB cable connector, see the lower black arrow in the picture.
Resulting support material when sliced:



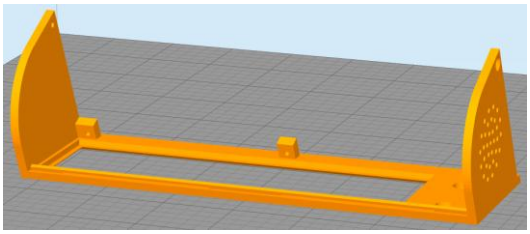


Bottom plate:
Orientation as shown.
No support material needed.

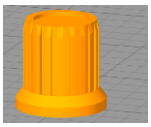


Copyright © 2022, Henk Beekhof All rights reserved.

Front plate:
Orientation as shown.
No support material needed.



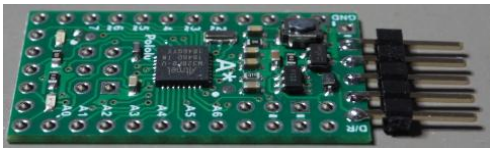
Rotary knob:
Orientation as shown.
No support material needed.



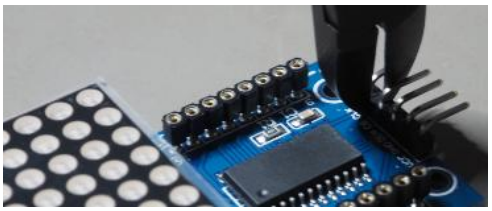
3.2 Step 2: Schematic, mount electrical components

This step describes in detail how to mount and connect the electronics. You might as well use the schematics and quicky check the pictures about the intention, basically these three items are not seen in the schematics:
1) How to mount the programming connector on the CPU board
2) The removal of the LCD connector
3) The replacement of the crystal on the RTC board

Prepare the CPU board
Solder a five pin straight header on top of the solder islands as shown in the picture. The purpose is to keep the bottom side of this PCB as flat as possible, therefor this soldering method is used. The two most upper terminals are GND. The upper most terminal is not connected.



Prepare the LCD board
If your LDB board contains the connector as shown in the picture on the right most side, then remove this connector.
If you do not have a desolder tool, just cut the connector to separate each pin and desolder the individual pins.

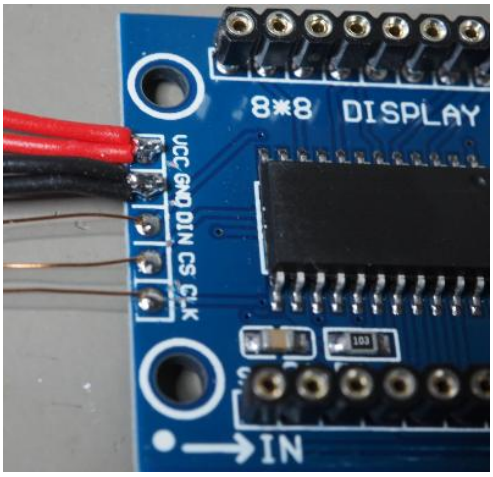


Where the connector is removed, the “IN” side of the PCB, wires are connected as shown in the picture.

Use two red wires for Vcc (5V), and two black wires for GND.

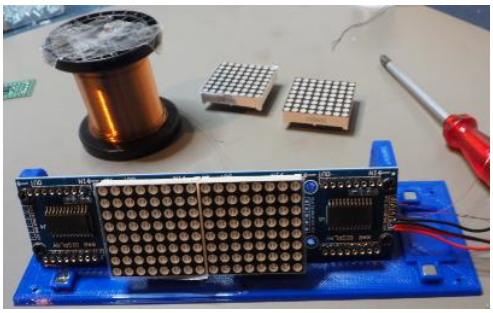
For the Din, CS and Clk pin a thin copper wire is used, which is isolated by a thin coating. This is the same sort of wire as what is used in transformer coils. Heat the end of the wire to burn away the coating. These thin wires are extremely handy in all kind of applications.

Make sure the bottom of the PCB remains flat, if solder is found on the other side if the pins, then cut it off with a knife.



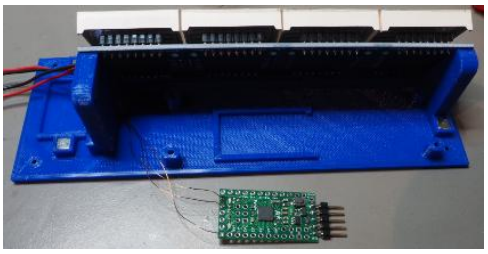
Copyright © 2022, Henk Beekhof All rights reserved.

Mount the LDC as shown in the picture.



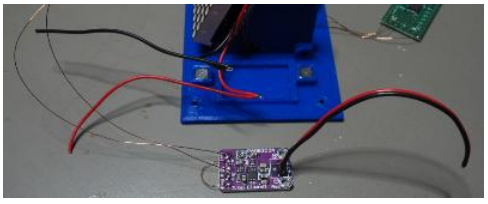
Mount the cubic magnets in the four square holes of the bottom case before you replace the two 8x8 LCD modules in the left side and right side of the LCD board.

Note: Let the magnets fall on a metal item, detach them and use that orientation to mount them in the bottom case. Use something sturdy like a screw driver to push them to the bottom of the hole.



Connect the LCD signal wires (Din, DS, Clk) to the CPU board as indicated by the schematic diagram.
Example: CS is connected to PB2. See chapter 7.1 and search for PB2.
Place the CPU in the correct orientation as shown in the picture, later, when all connections are done the CPU pcb will be pushed into the bottom plate in the 'island' shown in the picture just above the PCB.



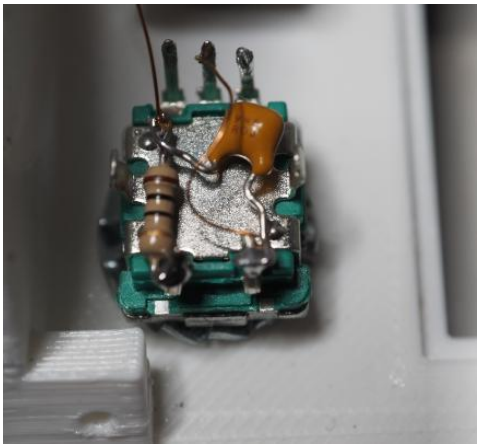


Connect wires to the audio amplifier board
Thin copper wires for A+ and SD.
Thin copper wire to connect A- with GND.
Normal wires for the speaker.
Strip the GND wire from the LCD and connect it to GND, leave a black wire as shown in the diagram to connect to the rotary.
Strip the Vcc wire from the LCD and connect it to Vcc. The diagram shows that the red wire is extended, this is not needed, this should be an 'end point', because the other Vcc from the LCD will be connected to the power switch to power both the LCD and Audio board with 5V.



Push the audio board in its placeholder, orientation as shown in the picture.

Mount the power switch and rotary in the front cover.



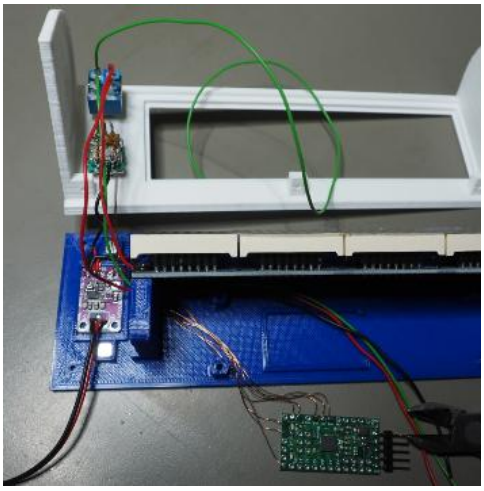
The 'bottom' side of the rotary contains the two switch terminals.
The right terminal is later connected to GND. First mount the 0.1μF capacitor and a thin wire to this terminal. Connect the thin wire (GND) to the middle pin of the three rotary terminals.

The other side of the capacitor is connected to the 10Ω resistor and a thin wire which actually is the 'switch' signal. Connect this wire to the CPU board. The other side of the resistor is connected to the left terminal of the switch.

Connect two thin wires to the upper left and right terminal of the rotary. These are the rotary signals. Connect these to the CPU board.

Copyright © 2022, Henk Beekhof All rights reserved.

Power wires
Connect the black (GND) wire from the audio board to the bottom right pin of the rotary (where the capacitor is connected to).



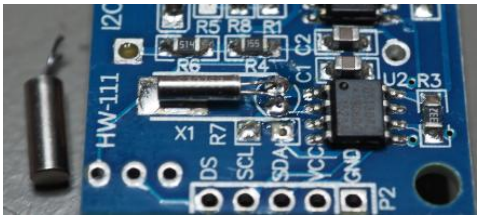
From the 6 pins the upper 4 are used, left is for 3V3 and right is for 5V.

Connect the red (5V) wire from the LCD to the upper right pin of the power switch. This pin powers the LCD and amplifier.

Connect a red wire to the middle right pin, this is the 5V source from the battery, guide the wire to the battery location. Do the same for the 3V3 source with a green wire. Together with the black wire from the LCD, these wires are found in the picture beneath the nippers.

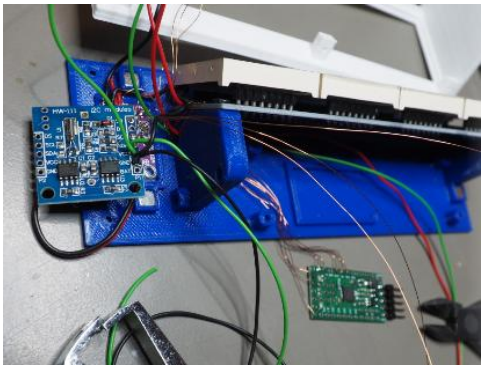
Connect a green wire to the upper left pin, this is later used to connect to the Vcc of the RTC and CPU.

The RTC board
DS1307 NOT RECOMMENDED TO USE:
The DS1307 manual specifies a crystal with C_L (Load Capacitance) of 12.5pF. I tested with crystals with different load specs, and it turned out that this influenced the accuracy heavily. My conclusion is that the boards which I got does not have a crystal with the required load capacitance. *Advice*: Replace the crystal with a crystal where you know for sure that the internal load capacitance is 12.5pF. Or use DS3231.



DS3231 board
This board is used and works as expected.
It can be located in the same position in the case, but should be glued.
The case was designed for the DS1307 board (mounting holes).

The RTC board will be mounted with an M3 bolt through the hole of the left LDC ‘holder’ shown in the picture.

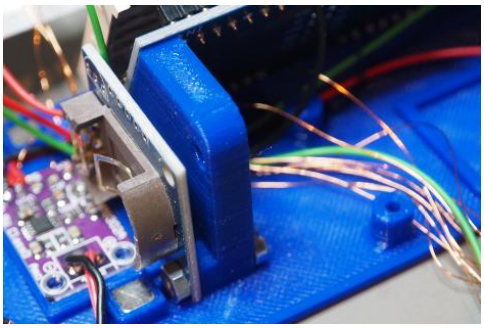


First connect thin wires to the SCL, SDA and SQ pins.

Connect the green wire from the switch to Vcc and let it continue.
Connect the black wire from the LCD to the GND and let it continue.

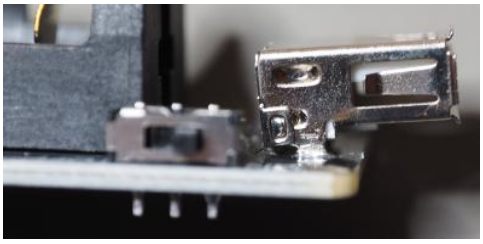
Bring these 5 wires to the battery/CPU side.

Mount the RTC board as shown in the picture.
See the note on page 4, remove resistor U4.
Mount the backup battery CR2032.



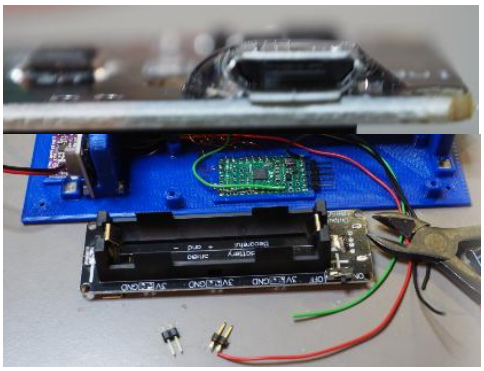
Copyright © 2022, Henk Beekhof All rights reserved.

The battery board contains a large USB connector and a small USB connector. Remove the large USB connector.
Easiest way to destructively demount the connector: You can displace it by heating the mechanical soldered connection one by one to lift it up, then cut through these two pins. Now you can move the connector back and forth until the USB pins at the back of the connector break.



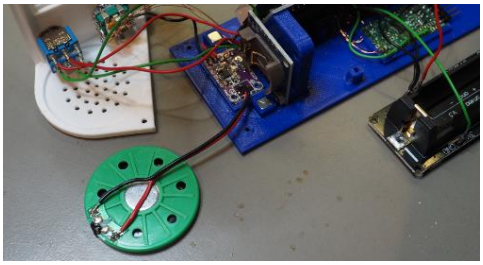
The micro-USB connector has two mechanical mounts. This makes this component the critical part of this board. I would recommend suppliers to use connectors with 4 mechanical mounts. To make it more robust I glue this connector. Make sure no glue gets into the connector.

The picture shows the orientation of the battery board. Mount for each supply (3V3 and 5V) a two pin header.



Connect the green 3V3 wire to the 3V3 supply.
Connect the red 5V wire to the 5V supply.
Connect the black wire from the RTC board and the black wire from the CPU board with GND.

Connect the speaker.
Make sure it fits mechanically in the ‘hole’ of the case.
Mount the speaker in the case and use some glue to keep it there.

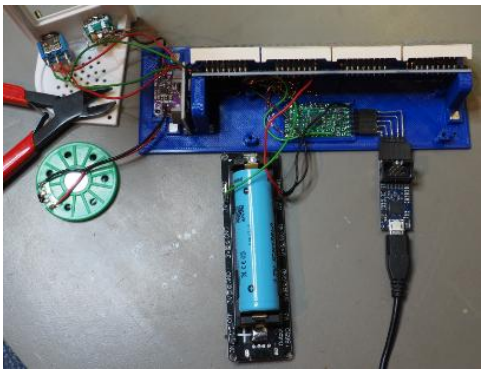


Everything is now nicely connected, ready for the next step:
Program and testing

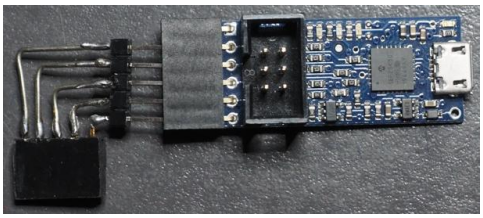
Copyright © 2022, Henk Beekhof All rights reserved.

3.3 Step 3: Programming and testing

For programming a Pololu board is used:
Pololu USB AVR Programmer v1.2
Reference: https://www.pololu.com/product/3172



I've soldered a '90 degree connector' to connect the programming board with the CPU. See second picture for detail, upper pin is not connected.



See chapter 4 to install the Arduino IDE and which settings to use to program this particular CPU board.

Power the board by the main switch.
Load the sketch provided, and program it in the board.
Result: The clock should be counting up.

Test the rotary knob, and audio: Press it once, if you turn it right the alarm time should increase. Set it to a low value and press the rotary again. The time should count down and an alarm should be heard when the time reaches 00:00:00.

Copyright © 2022, Henk Beekhof All rights reserved.

4 Arduino IDE

4.1 Installation

Via the link below you find information how to install the Arduino IDE on your system:
https://docs.arduino.cc/software/ide-v2/tutorials/getting-started/ide-v2-downloading-and-installing

This application was developed using version: 1.8.5. Later installed version: 2.0.3
Note: You might also consider to use Microsoft Visual Studio, this is for free, and is more advanced than the Arduino IDE related to ease of use, debugging features, this is a professional environment.

4.2 Location of the code + way of working

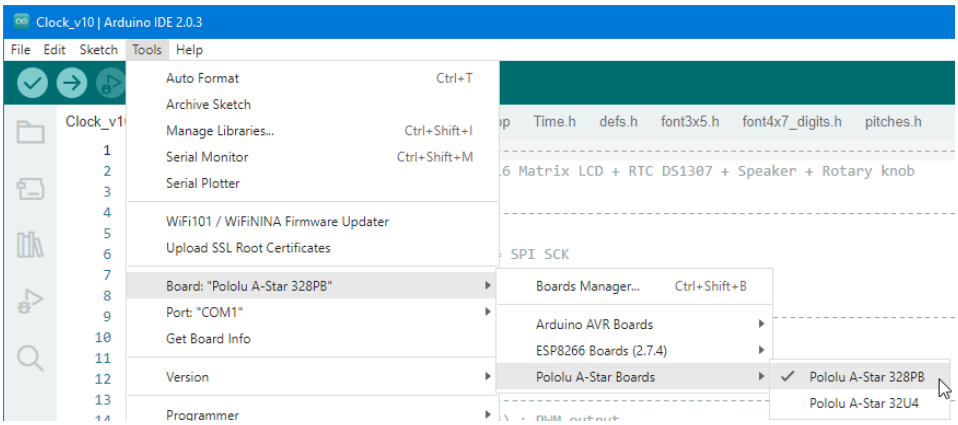
By default, the Arduino IDE stores the projects in c:\Documents\Arduino\
In File -> Preferences you can change this default path.

The (to be) developed code is not checked-in and checked-out by using a revision system, for now the way of working is to save a new project on a major new step in the development process. The Arduino IDE makes this very easy, since a 'Save as' saves all files of the project into the new project folder.
Method: Use 'Save as', move up to the projects folder, define a new name, which is the new project folder.

Differences in Arduino code related to other systems:
A project has two procedures, called setup() and loop(), after a powering on the Arduino board, first the setup() is executed once, then the loop() is started, which runs repeatedly.

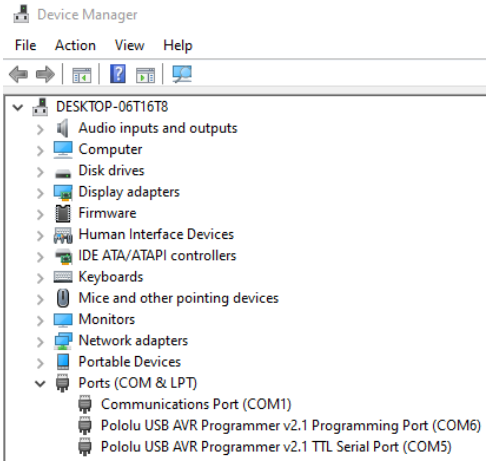
4.3 Setup Board and Port to compile and upload software

Step 1: Define the target board
In Arduino IDE: Tools -> Board: -> Select the applicable board.
Note: This is an IDE setting, thus not related to the project file (.ino), which is strange....and sort of inconvenient if you do parallel developments using different boards.



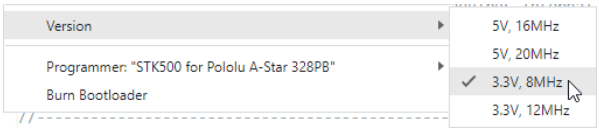
Step 2: Define the serial port
Start the Device Manager and open the section 'Ports' to find out which port is used.
Search for the item "Pololu USB AVR Programmer v2.1 TTL Serial Port (COMx)"

Copyright © 2022, Henk Beekhof All rights reserved.

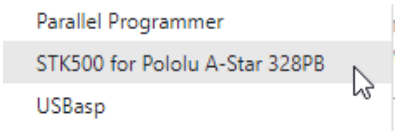


In Arduino IDE: Tools -> Port: Select the correct COM port found in the Device Manager, (COM**5**)

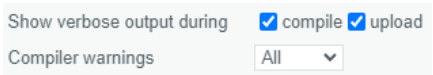
Step 3: Set the board version
In Arduino IDE: Tools -> Version: Select the 3.3V, 8MHz option



Step 4: Define the programmer
In Arduino IDE: Tools -> Programmer -> Select "STK500 for Pololu A-Start328PB"
NOTE: In newer versions (tested in v2.3.6) of the IDE and Board related code this option should not be used, use instead “Atmel STK500 development board”.



Step 5: Optional: Change settings to verbose output
In Arduino IDE: File -> Preferences



Step 6: Optional: Test if the settings are correct by compiling and uploading a simple sketch
If you want to test these settings then write a simple program. For instance:

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  while (1) {
      Serial.println("Ok");
  }
}
```

Press the Upload button which also compiles the code if not yet done:
Then open the Serial Monitor, that should show lots of 'OK's. Tools -> Serial Monitor or keys: Ctrl + Shift + m.

 Copyright © 2022, Henk Beekhof All rights reserved.

5 The software

5.1 Library dependencies

The software depends on three libraries:
- MD_MAX72xx
- AT24CX
- RTCLib

In the source code folder the libraries are included in folder "project_libraries".
These are for reference only, these files are not included in the software from this location.

These three folders should also be found in the library folder: C:\Users\User\Documents\Arduino\libraries
They are used from that location by the Arduino IDE.

The RTClib library and MD_MAX72xx library is found in the library manager.

The AT24Cx library is not found in the library manager, download it from github (https://github.com/cyberp/AT24Cx) as a zip file and install that zip file (Sketch -> Include library -> Add .ZIP library...).

It seems not very straight forward within the Arduino environment to use libraries from 'your own' specified location. This makes it hard to distribute independent code or make a backup copy of your code which includes all dependencies.

It is the way it is, thus please either install these libraries or just copy the directories from the "project_libraries" into the Arduino library folder.

5.2 Operation

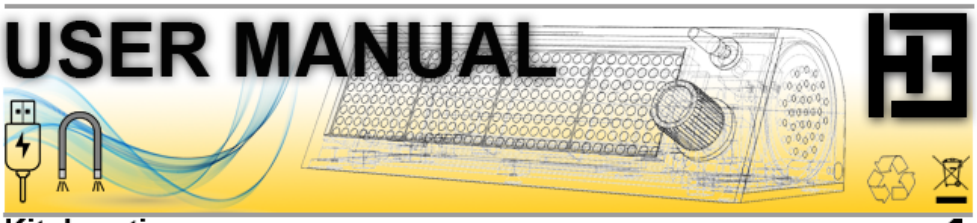
The basic operation is simple. The RTC is programmed to generate a 1Hz pulse on the SQ pin. This pin is connected to an interrupt input of the CPU. On each 1Hz interrupt the 'interrupt_1Hz_flag' is set.
The loop() function checks this flag, when set it updates the time and resets the flag. The clock has a 'dark mode' which adds some code to this part.

Then the loop() function checks if the rotary button is pushed, in 'dark mode' this turns on the display. Further it checks if the button is pressed once or twice. If it is pressed once, then the kitchen clock mode is entered. If it is pressed twice, the menu is entered.

For more details: Check the comments in the code and check the operating manual.

Copyright © 2022, Henk Beekhof All rights reserved.

6 Operating manual



Kitchen timer 1

↓ Show timer Time ↻ Increment or decrement timer Time ↓ Store Time & Start timer

Press the knob once to activate the timer function. The last stored timer duration is shown.
This timer duriation can be increased or decreased by rotating the knob.

Press the knob to store the selected timer duration, and to start the timer. The time counts down,
when zero is reached the timer plays a tune. In the settings you define which tune is played.
The timer alarm stops when you press the knob. The timer function exits and the current time is shown.

Stopwatch 2

↓↓ ↻ Menu: Stopwatch ↓ 00:00:000 ↓ Start stopwatch ↓ Stop, Show elapsed time ↓ Exit

Press the knob twice to show the menu.
Rotate to select "Stopwatch" and press the knob to enter the stopwatch function.
The display shows 00:00:000, this indicates minutes, seconds and thousands of seconds.
When the knob is pressed the stopwatch starts counting.
When the knob is pressed again the stopwatch stops, and the elapsed time is shown.
Press the knob to return to the clock function.

Alarm 3

↓↓ ↻ Menu: Alarm ↓ ↻ On ↓ Set & Exit
↻ Off ↓ Set & Exit
↻ Time ↓ ↻ Hours ↓ Set ↻ Minutes ↓ Set ↻ Seconds ↓ Set & Exit
↻ Tune ↓ ↻ S1, S2, S3, (play tune) ↓ during play Stop tune Return to S menu
↓ after play Set & Exit
↻ Count ↓ ↻ 1..250 select alarm repeat ↓ Set & Exit

Press the knob twice to show the menu.
Rotate to select "Alarm" and press the know to enter the "Alarm menu".
The alarm plays a user selectable tune at the specified time.
When the red dot in the upper left corner of the display is 'on' this indicates the alarm function is turned on.

Settings 4

↓↓ ↻ Menu: Settings ↓ ↻ Tune ↓ ↻ S1, S2, S3, (play tune) ↓ during play, Stop tune, Return to S menu
↓ after play, Set & Exit
↻ Time ↓ ↻ Hours ↓ Set ↻ Minutes ↓ Set ↻ Seconds ↓ Set & Exit
↻ Date ↓ ↻ Year ↓ Set ↻ Month ↓ Set ↻ Day ↓ Set & Exit
↻ LCD ↓ ↻ 1, 2, 3, ... , 15 Brightness (1=dimmed, 15=bright) ↓ Set & Exit
↻ Dark ↓ ↻ On : Display turned off, each ↓ shows time for 10 sec's ↓ Set & Exit
↻ Off : Display on ↓ Set & Exit

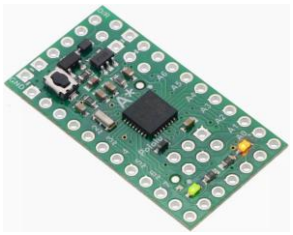
Press the knob twice to show the menu.
Rotate to select "Settings" and press the knob to enter the "Settings" menu. Here you can specify:

- The tune which is used by the kitchen timer.
- The current time.
- The current date.
- The LCD brightness.
- The Dark mode.

Copyright © 2022, Henk Beekhof All rights reserved.

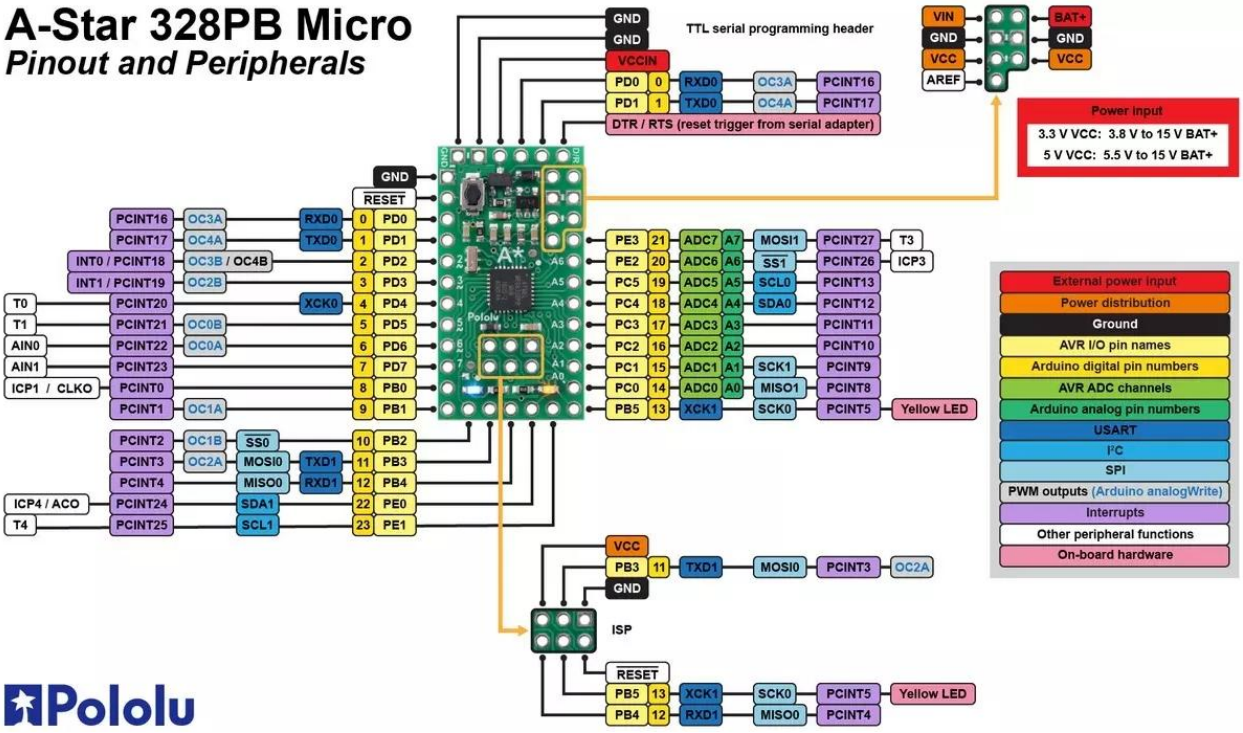
7 Detailed info - HW boards

7.1 A-Star 328PB Micro and USB AVR Programmer v2.1



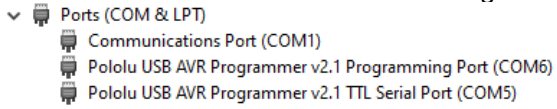


Reference: https://www.pololu.com/docs/0J74/all (CPU)
https://www.pololu.com/product/3172 (Programmer)



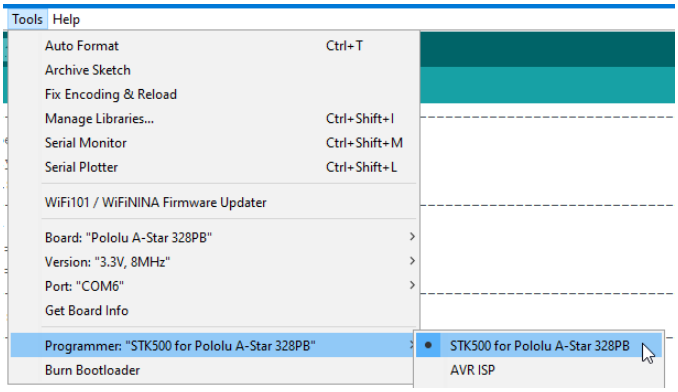
Programming settings in Arduino IDE (Menu: "Tools"):
- The board should be set to "Pololu A-Star 328PB"
- The version should be set correctly depending on the used board
- The port should be set to the port 'n' connected to "....Programming Port (COM n)"
- The programmer should be set to "STK500 for Pololu A-Star 328PB"
- The programming board does not supply the CPU board, thus provide a power supply to the CPU board

Use the standard windows Device Manager to find the used "Programming Port" COM port

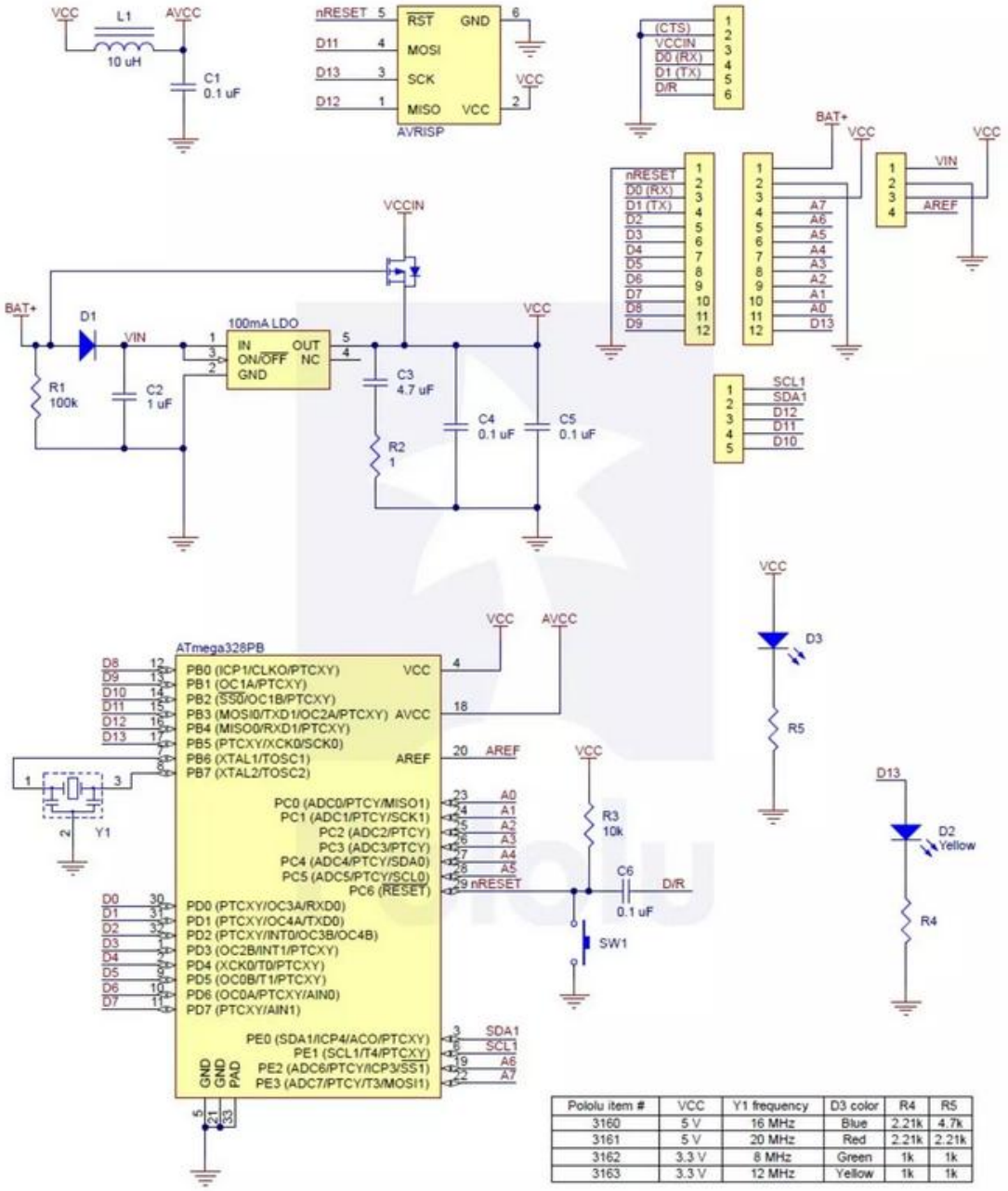


Screenshot of settings:

 Copyright © 2022, Henk Beekhof All rights reserved.



Schematic CPU board: (Including table D3 color led indicates the board type related to supply and frequency)

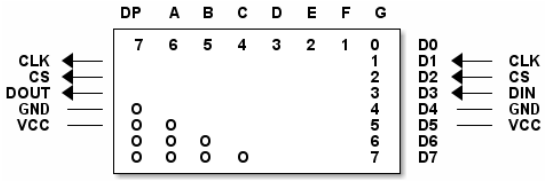


Pololu item #	VCC	Y1 frequency	D3 color	R4	R5
3160	5 V	16 MHz	Blue	2.21k	4.7k
3161	5 V	20 MHz	Red	2.21k	2.21k
3162	3.3 V	8 MHz	Green	1k	1k
3163	3.3 V	12 MHz	Yellow	1k	1k

Copyright © 2022, Henk Beekhof All rights reserved.

7.2 FC16 Matrix LCD (MAX7219)





Module:	FC16
IC:	MAX7219
LCD:	1088AS
8x8LED	
Vcc:	5V
Control:	SPI

Columns and Rows
Asuming the input cable is positioned at the right side, the the following info about columns and rows is valid:
In above diagram the X axis represents Column, starting at 0 at the right side.
Columns: G=0, F=1, DP=7, and the next chained segment G=8, F=9, etc.
Rows: D0 to D7 represent a row. D0 is the LSB of the data.
The triangle of dots in the bottom left corner, the dot in 'coordinate' {DP,D4} equals {Col=7,Row=4}

Fonts
Byte 0: The number of column bytes (N) that form this character, equals zero if the character is undefined
Byte 1..N: One byte for each column of the character. Thus it represents the content of a column. The LSB of this byte corresponds to row 7, the MSB corresponds to row 0.

Example: Character '&', represented in 5 bytes:
5, 0x36, 0x49, 0x56, 0x20, 0x50, // 38 - '&'

Column	9	8	7	6	5	4	…	0
Font Byte	1	2	3	4	5			
Row 0, bit 7		■						
Row 1, bit 6	■		■					
Row 2, bit 5	■		■					
Row 3, bit 4		■						
Row 4, bit 3	■		■		■			
Row 5, bit 2	■			■				
Row 6, bit 1		■	■		■			
Row 7, bit 0								
Font value [hex]	36	49	56	20	50			

Library MD_MAX72xx

Font definition
The font is located in file: **MD_MAX72xx_font.cpp**
This is a variable sized font, the first byte of a character specifies the width

```
MD_MAX72XX::fontType_t PROGMEM _sysfont_var[] =
{
  'F', 1, 0, 255, 8,
  0,  // 0 - 'Empty Cell'
  5, 0x3e, 0x5b, 0x4f, 0x5b, 0x3e,  // 1 - 'Sad Smiley'
  5, 0x3e, 0x6b, 0x4f, 0x6b, 0x3e,  // 2 - 'Happy Smiley'
  5, 0x1c, 0x3e, 0x7c, 0x3e, 0x1c,  // 3 - 'Heart'
```

See library documentation on the internet or from the downloaded library:
file:///C:\Users\User\Documents\Arduino\libraries\MD_MAX72XX-master\docs\index.html
Examples to set a point or a byte on a row or column:

```
setPoint(Row, Col, Value);
setRow(RowNr, Value);
setColumn(ColumnNr, Value);
```

How to position a character
setFont(fontType_t* f);

Copyright © 2022, Henk Beekhof All rights reserved.

Other definition of fonts: Using a fixed size
For the width of characters a fixed size is used. Setting a pointer to a character is easy, however based on the font definition the order of characters needs to be handled.

This specifies a font named `myfont`, containing 80 characters, where each characters is 5 bytes wide.

```
unsigned const char PROGMEM myfont[80][5] = {
  {0, 0, 0, 0, 0},                  // Space
  {0x3f, 0x48, 0x48, 0x48, 0x3f},   // A
  {0x7f, 0x49, 0x49, 0x49, 0x36},   // B
```

This requires mapping of ascii character value to a position in the font table, example:

```
if (c >= 'A' && c <= 'Z' ) {
    c &= 0x1F;                  // A-Z maps to 1-26
} else if (c == ' ') {
    c = 0;                      // space is located on position 0
} etc. etc.
```

This is one standard mapping routine, the resulting value for c is used in next fetch action:

```
for (char col = 0; col < 5; col++) {
  fontByte = pgm_read_byte_near(&myfont[c][col]);
```

Parola library
See library documentation on the internet or from the downloaded library:
file:///C:/Users/User/Documents/Arduino/libraries/MD_Parola-master/docs/index.html

Definition and instantiation

```
#include <MD_Parola.h>
#include <MD_MAX72xx.h>
#include <SPI.h>

// Define the matrix LCD type and size
#define HARDWARE_TYPE MD_MAX72XX::FC16_HW
#define MAX_DEVICES 4
#define CLK_PIN   D5
#define DATA_PIN  D7
#define CS_PIN    D8

// Define the display
MD_Parola P = MD_Parola(HARDWARE_TYPE, CS_PIN, MAX_DEVICES);
char curMessage[64];
void setup() {
  P.begin();
  P.displayClear();
  P.print("Text");
  Delay(5000);
  P.displaySuspend(false);
  strcpy(curMessage,"This is text");
  P.displayScroll(curMessage, PA_LEFT, PA_SCROLL_LEFT, 25);
}
void loop() {
  if (P.displayAnimate()) {
    P.displayReset();
  }
}
```

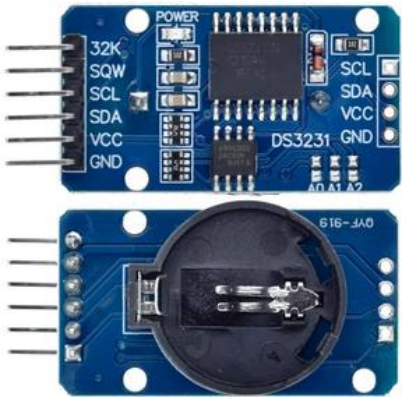
Display a character at a specific location
This is not possible with this library. It handles strings, and you can devide the display in several zones, however a zone boundary is a module boundary. It is not possible to create a zone from a certain column to another column. This makes the usage of zones limited and for instance not usable when you want to update specific characters, such as in a counter or time or date.

Writing the whole display over and over again has on the other side a benefit: Disturbance could set unwanted pixels, they stay when that part of the display is not updated.

Thus when individual character positioning is needed, one could consider to use the MD_MAX72xx library without the MD_Parola library.

Copyright © 2022, Henk Beekhof All rights reserved.

7.3 RTC/EPROM Module DS3231 / AT24C32



VCC = 3.3V to 5V
SQW = Square Wave

RTC:

Chip: DS3231

Note: When battery is used remove U4 or the diode next to it.

The RTC is used through the RTClib:
https://adafruit.github.io/RTClib/html/class_r_t_c___d_s1307.html

For convenience I have wrapped this library in a Time_Interface class. This is extremely handy when the underlying hardware changes, actually this class was used on other hardware first.

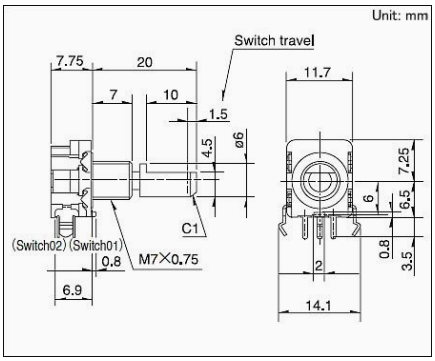
EEPROM

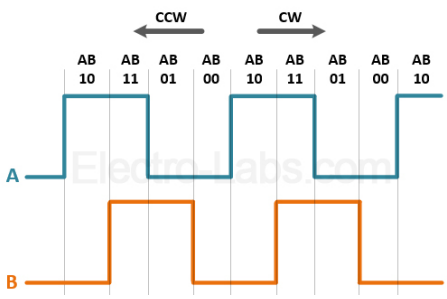
Chip:	AT24C32, 32K := 4096 x 8 = 4K bytes
Endurance:	1 million write cycles
Write:	32 Byte page write mode (partial page writes allowed)
I2C Address EPROM:	0x57 (Note: the default address on this board is different from the inaccurate RTC board)

eeprom library: https://github.com/cyberp/AT24Cx

7.4 Rotary encoder + switch







Copyright © 2022, Henk Beekhof All rights reserved.