Motion Controlled Matrix Bed Clock

Created by Flavio Fernandes

https://learn.adafruit.com/motion-controlled-matrix-bed-clock

Last updated on 2023-08-29 03:55:50 PM EDT
# Table of Contents

## Overview
- A multi-function display by your bedside!
- Prerequisite Guides
- Parts

## Circuit Diagram

## 3D Printing

## Coding
- Customization Options
- Enable I2C via raspi-config
- Install Matrix Driver
- Testing the Matrix Code
- Install Git, Pip and Clone
- Automatic Startup
- Customize tweaks
- MOTT
- Remove/disable unnecessary services
- Safely Shutdown setup

## Assembly
- Prep Bonnet
- Jumper
- Bridge Pad
- Attach LED Matrix to Frame
- Glue on!
- Solder APDS9960
- Hang overboard
Overview

A multi-function display by your bedside!

The 64x64 LED Matrix is a great platform to display anything you care about. Like the time and what day it is. :) With the motion and light sensor, the display can be made to dim or blank out so it will not disturb you while sleeping. A simple wave motion in the dark room is enough to re-enable the display for a short period.

The 3D printed harness file() is available with this project, so you can adapt it to suit your bed or desk space needs. In the current form, two printed parts are glued together with regular table pin connectors. You will also need four M3x5mm flat head screws in order to connect the matrix display to the printed frame.

The code is written in Python and has lots of potential for easy improvements. The current implementation gives you the ability to connect to an MQTT() broker for subscribing and publishing events in a canonical IOT() setup. But you need not to worry about that if a clock is all you care about. Underneath the Python code, this project uses the display driver written by Henner Zeller(), which makes the display fast and stable when used together with the Adafruit parts.

Prerequisite Guides

I suggest walking through the following guides to get a better understanding of the electronics.

- Adafruit APDS9960 breakout Digital Proximity, Ambient Light()
- RGB Matrix Bonnet for Raspberry Pi()
### Parts

<table>
<thead>
<tr>
<th>Item</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 x Raspberry Pi Zero WH (Zero W with Headers)</td>
<td><a href="https://www.adafruit.com/product/3708">https://www.adafruit.com/product/3708</a></td>
</tr>
<tr>
<td>Offers all the benefits of the Pi Zero W and includes a pre-soldered GPIO header</td>
<td></td>
</tr>
<tr>
<td>1 x Adafruit RGB Matrix Bonnet for Raspberry Pi</td>
<td><a href="https://www.adafruit.com/product/3211">https://www.adafruit.com/product/3211</a></td>
</tr>
<tr>
<td>Create a dazzling display with your Raspberry Pi with the Adafruit RGB Matrix Bonnet.</td>
<td></td>
</tr>
<tr>
<td>1 x 64x64 RGB LED Matrix</td>
<td><a href="https://www.adafruit.com/product/3649">https://www.adafruit.com/product/3649</a></td>
</tr>
<tr>
<td>2.5mm Pitch - 1/32 Scan</td>
<td></td>
</tr>
<tr>
<td>1 x Adafruit APDS9960 Proximity, Light, RGB, and Gesture Sensor</td>
<td><a href="https://www.adafruit.com/product/3595">https://www.adafruit.com/product/3595</a></td>
</tr>
<tr>
<td>This breakout is chock full o' sensors!</td>
<td></td>
</tr>
<tr>
<td>1 x 5V 10A switching power supply</td>
<td><a href="https://www.adafruit.com/product/658">https://www.adafruit.com/product/658</a></td>
</tr>
<tr>
<td>5V DC up to 10 Amps, running from 110V or 220V power</td>
<td></td>
</tr>
<tr>
<td>1 x SD/MicroSD Memory Card (8 GB SDHC)</td>
<td><a href="https://www.adafruit.com/product/1294">https://www.adafruit.com/product/1294</a></td>
</tr>
<tr>
<td>8 GB class 4 micro-SD card. It comes with a SD adapter.</td>
<td></td>
</tr>
<tr>
<td>1 x USB to TTL Serial Cable - Debug / Console Cable for Raspberry Pi</td>
<td><a href="https://www.adafruit.com/product/954">https://www.adafruit.com/product/954</a></td>
</tr>
<tr>
<td>The cable is easiest way ever to connect to your microcontroller/Raspberry Pi/WiFi router serial console port</td>
<td></td>
</tr>
</tbody>
</table>

### Circuit Diagram

Take a moment to review the components in the circuit diagram. This illustration is meant for referencing wired connections - the length of wire, position and size of components are not exact.

Note that in order to drive the 64x64 dot matrix display, you must connect GPIO pins 4 and 18.

RGB Matrix Bonnet connects to the Raspberry Pi Zero.

The 2.1mm Male DC adapter and will need to be 150mm long.
While it is okay to use a single power supply to feed both RPI and matrix display, it is important that the throughput is powerful enough.

5v 10A switching power supply is plenty for this project.

3D Printing

The OnShape online tool was used in order to design the structure used for the bedclock. It exports STL files that you can use for 3D printing.
Note: A huge thanks for making this part of the project possible goes to Brian Bailey (), my friendly CAD guru.

By printing this project in multiple parts, there is no need for printing support fillers. Since the parts have nice cavities for the table pins, you will have no problems aligning and binding them together.

There is a third and smaller part for covering the RPI on the back side of the frame. You can see and print that from thingiverse.com ().

We used regular PLA (Polylactic Acid) filament, which is widely common.
Once printed, have these parts connected using table pins as shown here.

More details will be provided on the assembly section, but here is a picture showing how the 3 main parts go together:

- Top bezel
- Front bezel
- Matrix display
Coding

First, make sure you've loaded the latest Raspbian Lite operating system on your Raspberry Pi. You can find the OS image download here, instructions for burning to an SD card image here, and a convenient guide here that explains how to load an operating system.

Next, have your Raspberry Pi connected to the Internet, either with a wired connection to its ethernet port, or by setting up wireless access to a WiFi network. Check out the guide on network setup for more details on using wireless and WiFi networks with the Pi.

Once your Raspberry Pi is powered up and connected to a network you can follow the steps below to install the bedclock software.
If you’re familiar with connecting to the Raspberry Pi over SSH you can use an SSH terminal application to connect. If you aren’t familiar with SSH, you can use the Adafruit Pi Finder tool to find your Raspberry Pi and open a terminal to run the installation. In either way, you should be able to be logged into the RPI as user pi.

### Bedclock Code

Note with the latest versions of Raspbian (Jessie full & lite since ~February 2017) they disable SSH by default! Read below to see how to enable SSH so you can access the Pi with tools like Pi Finder.

### Customization Options

As part of the Raspbian installation, visit the various options available and see if any of them needs tweaking. Setting the timezone is a good candidate. :)

### Enable I2C via raspi-config

In order to read the gesture sensor, I2C must be enabled via the raspi-config interface. Follow the guide below for a step by step guide. If your familiar with this, use sudo raspi-config and enable I2C through the “Interfacing Options.”

```bash
sudo raspi-config
```

- Localization Options -> change timezone
- Interfacing Options -> enable ssh
- Interfacing Options -> enable I2C

### Raspberry Pi – Enable I2C Guide

```bash
sudo apt-get install -y python-smbus i2c-tools
# Ensure i2c device is present
ls /dev/i2c* && echo ok || echo bad error cannot see i2c
```

### Install Matrix Driver

OK, now you are ready to install the driver for the LED matrix. We have a script that downloads the code and any prerequisite software. It works with the current Raspbian
“Stretch” operating system (either the Lite or Desktop version). Walk through the options it presents and select the ones that pertain to your setup. If you get stuck, we have a more details in the [Bonnet Matrix guide](#).

```bash
sudo bash rgb-matrix.sh
```

The script will confirm your selections and offer one more chance to cancel without changes.

There’s a lot of software to update, download and install, so it may take up to 15 minutes or so to complete. Afterward, you’ll be asked whether you want to reboot the system.

## Testing the Matrix Code

After the reboot, it’s time to try the display! Run the following command and make sure it is working properly. By the way, it is okay to do this after the whole assembly; just come back here once the display is properly powered and connected.

```bash
cd ~/rpi-rgb-led-matrix/examples-api-use
ROTATE=180
TIMEOUT='t 10'
for X in 0 3..11 ; do \\
    PARAMS='--led-rows=64 --led-cols=64 --led-brightness=88'
    sudo ./demo ${PARAMS} ${TIMEOUT} --led-pixel-mapper="Rotate:${ROTATE}" -D${X}
    sleep 0.3
done
```

Press Control+C to stop the program and get back to the command line.

## Install Git, Pip and Clone

Let’s get the git command so we can clone the bedclock software repository. We will also need some Python packages, which we can easily obtain by using the pip command.

```bash
sudo apt-get install -y python3-pip git
BRANCH=adafruit
cd &;&;&;
git clone -b ${BRANCH} \\
https://github.com/flavio-fernandes/bedclock.git bedclock.git

cd ~/bedclock.git &;&;&; sudo -H pip3 install --upgrade -r ./requirements.txt
```

©Adafruit Industries
Automatic Startup

We will use `systemd` to start the program automatically. This is just a copy command away from the cloned repo.

```
# add bedclock service to systemd
sudo cp -v ~/bedclock.git/bedclock/bin/bedclock.service /lib/systemd/system/
sudo systemctl enable bedclock.service

# start the application
sudo systemctl start bedclock.service
```

You can monitor what is happening with the app by looking at the log or systemd status:

```
sudo systemctl status bedclock.service
tail -F ~/bedclock.git/syslog.log | grep bedclock
```

Customize tweaks

There are lots of knobs that can be tweaked to make the clock behave as you want. Most of these settings are located in the file called `const.py`. After making the changes, make sure to restart the application by typing the following command:

```
sudo systemctl restart bedclock.service
```

If you want to start it manually, use `start_bedclock.sh`

```
sudo systemctl stop bedclock.service
~/bedclock.git/bedclock/bin/start_bedclock.sh
```

Just like the time you ran the matrix demo, press the Control-C keys to stop the program and get back to the command line.

MQTT

If you have an MQTT broker that your bedclock can talk to, you can enable it via the `mqtt_enabled` setting in the `const.py` file.

That file will be located under the following directory:
Edit file and modify the following parameters:

```makefile
# mqtt_broker_ip =
# mqtt_enabled = True
```

```
/bin/nano ~/bedclock.git/bedclock/const.py
```

Receiving events published by the bedclock application

```
# Install mqtt client to poke bedclock via mqtt broker
$ sudo apt-get install -y mosquitto-clients
$ mosquitto_sub -v -h ${MQTT_BROKER} -t "/bedclock/motion" -t "/bedclock/light"
/bedclock/motion on
/bedclock/motion off
/bedclock/light 365
```

Configuring the clock to stay awake in the dark

If you don’t want the clock to go blank when the room is dark, you can change its default behavior in `const.py` by setting `scr_stayOnInDarkRoomDefault = True`. Or via MQTT, as shown below:

```
$ mosquitto_pub -h ${MQTT_BROKER} -t /bedclock/stay -r -m "on"
$ mosquitto_pub -h ${MQTT_BROKER} -t /bedclock/stay -r -m "off"
# to clear topic from broker and use const.scr_stayOnInDarkRoomDefault
$ mosquitto_pub -h ${MQTT_BROKER} -t /bedclock/stay -r -n
```

Remove/disable unnecessary services

RPI Zero W handles the display just fine but it does not hurt to help it out by disabling unused things that compete for cycles. These were the ones I did on my setup, based on Henner’s advice:

```
sudo systemctl disable avahi-daemon &>&&
sudo systemctl stop avahi-daemon &>&&
sudo apt-get remove -y bluez bluez-firmware pi-bluetooth triggerhappy &>&&
echo ok
```

Safely Shutdown setup

Abruptly cutting power to the Pi can cause the files in the SD card to become corrupt. So, we suggest safely shutting down the Pi. There is a dedicated guide for enabling Read-Only mode, which will allow you to quickly turn off the Pi without risking corrupting the SD card.
Assembly

Prep Bonnet

First, we'll need to add a jumper to connect two pins and then bridge two pads to enable the RGB Matrix Bonnet to work with the 64x64 LED Matrix.

Jumper

We'll start by measuring a wire 30mm long. Tin and solder pin 4 to pin 18. Carefully bend the wire and move it away from the headers on the Bonnet.

Bridge Pad

Next, we need to turn the Matrix Bonnet over and locate the three solder pads. Tin the middle pad "E" and the "8" pad. Now heat up one of the pads and drag solder over to the next pad to bridge the two connections.
Attach LED Matrix to Frame

Once 3D frame is printed and glued, use flat screws to connect the display. It is important to keep these flat, to it sits seamlessly over the headboard surface.

The ribbon cable goes next and in order to make the surface flat, I went ahead and removed the back side of the plastic shield that attaches the end cable to the display.

Carefully solder 5V and GND to display, so it is... you get the idea... FLAT!
Glue on!
With glue gun hot and ready, I bonded the RPI and ribbon cable to the back side of the frame. There are plenty of holes in the back plane, so you can attach the RPI to the frame.

The cover provided as the 3rd STL file will fit nicely over the RPI, so this part is not exposed.
Solder APDS9960

The Proximity, Light, RGB, and Gesture Sensor is soldered to the Bonnet. Measure four wires long enough to connect 5v, GND, SDA, and SLC on the Bonnet and APDS9960.
Hang overboard

And that is it, folks! For more pictures on this assembly, take a look at this link().

You might consider a stand instead for nightstand use.

Check out this video() to see the bedclock in action