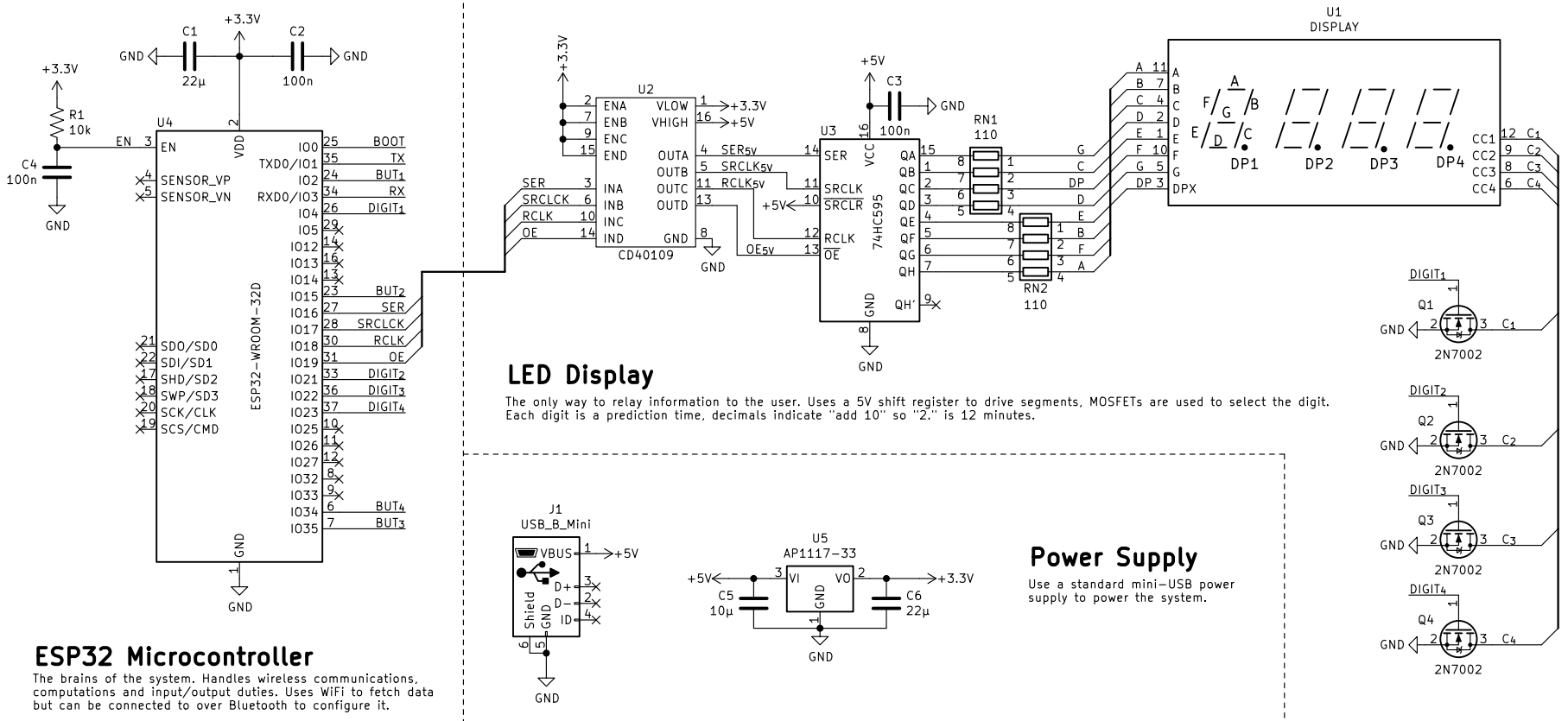


LED Display

The only way to relay information to the user. Uses a 5V shift register to drive segments, MOSFETs are used to select the digit.
Each digit is a prediction time, decimals indicate "add 10" so "2." is 12 minutes.

ESP32 Microcontroller

The brains of the system. Handles wireless communications, computations and input/output duties. Uses WiFi to fetch data but can be connected to over Bluetooth to configure it.

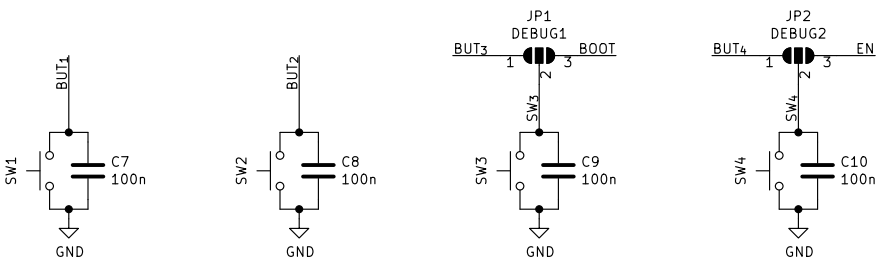
Power Supply

Use a standard mini-USB power supply to power the system.

User Buttons

User input buttons, one for each stop/route group.
For development/debugging buttons 3 and 4 can be rerouted to hasten programming.

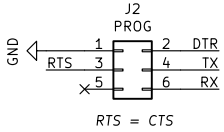
Alternate ("DEBUG") modes for switches 3 and 4:
3 – BOOT (Hold when restarting to program over serial)
4 – EN (Restart ESP32 when pressed)

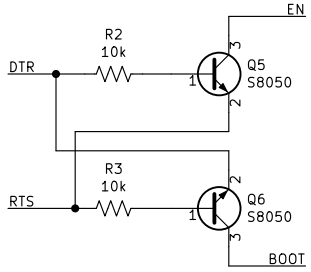


Programming

Programming interfaces and hardware to enable UART programming

DO NOT CONNECT TO PROG BEFORE SUPPLYING POWER!





Mounting Holes

H1 and H3 are 5.5mm from sides
H2 is centered. All are 7mm from the top edge.

H1 4MM H2 6.5MM H3 4MM

Board to display the next four predictions for TTC stops in minutes
Users interact using a button to select a group of interest
Users should be able to configure it over bluetooth with a phone
System is programmed using UART

Title: TTC Prediction Sign

Savo Bajic	Date: 2022-03-15	**Rev: 1.2**
KiCad E.D.A. eeschema 6.0.1-79c1e3a40b-116-ubuntu21.04.1	Size: USLetter	Sheet: 1/1

POSSIBLE IMPROVEMENTS:
- Add external pull up resistors for buttons and other lines that need pulling up (GPIO0?)
- Connect 3.3V to programming header
- Use some surface mount displays
- Add some status LEDs for the ESP32 to directly control
- Get better buttons (harder than 100g press)
- Move to Micro-USB or USB-C power
- Look into a better programming method.
  - Maybe try the continuous version with the modifications to the EN line?