Netrol

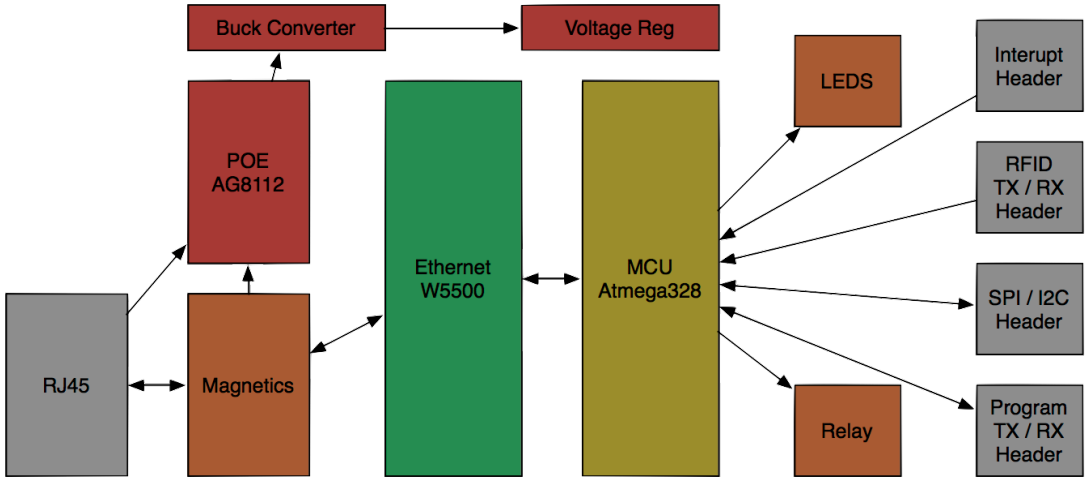
Project synopsis

Netrol short for Network Controller was designed as a cheep alternative to access control around a community premises. Having an Ethernet controller onboard allows the access control to be done on a server, this enables full automation and the ability to grant people access to the entire building or sections of without the need to visit each door and reprogram the access codes. The added feature of Power over Ethernet (PoE) creates an easier install process as power is sourced from the main Ethernet switch, this in turn creates a more secure premises when the switch is utilising a Uninterruptable Power Supply (UPS) as the doors will still be able to operate during a power failure.

The main functionality and features are

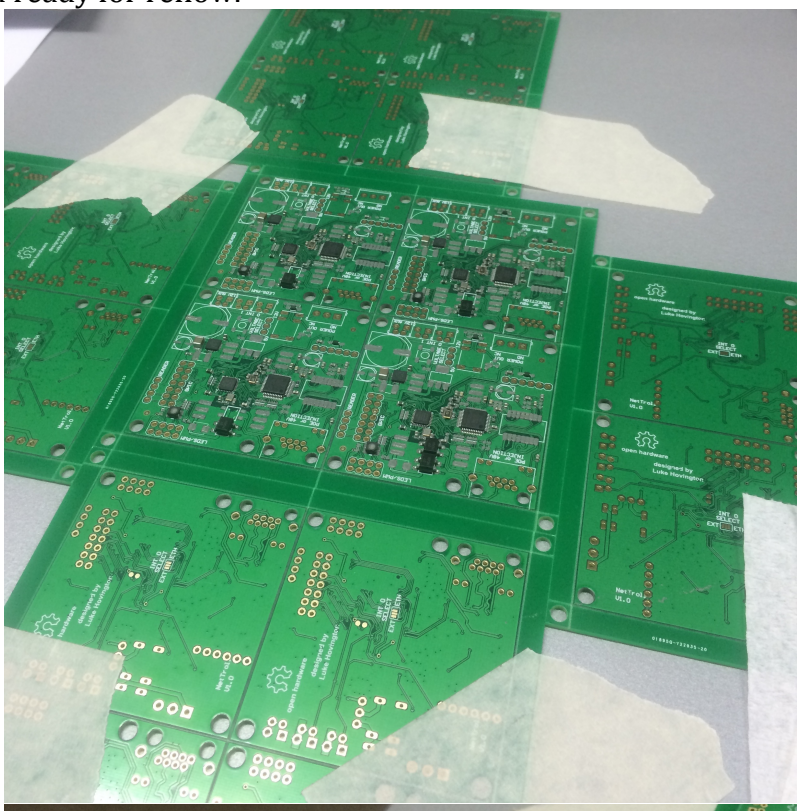
- Wiznet W5500 Ethernet controller for asking the server for information
- Arduino friendly Atmel atmega328
- Relay for switching door strikes or magnetic locks
- Silver Tech AG8112 Power over Ethernet (PoE) module
- Status via onboard LEDS
- SPI and I2C pins broken out for further expansion.
- Interrupt pins to enable press to exit buttons.

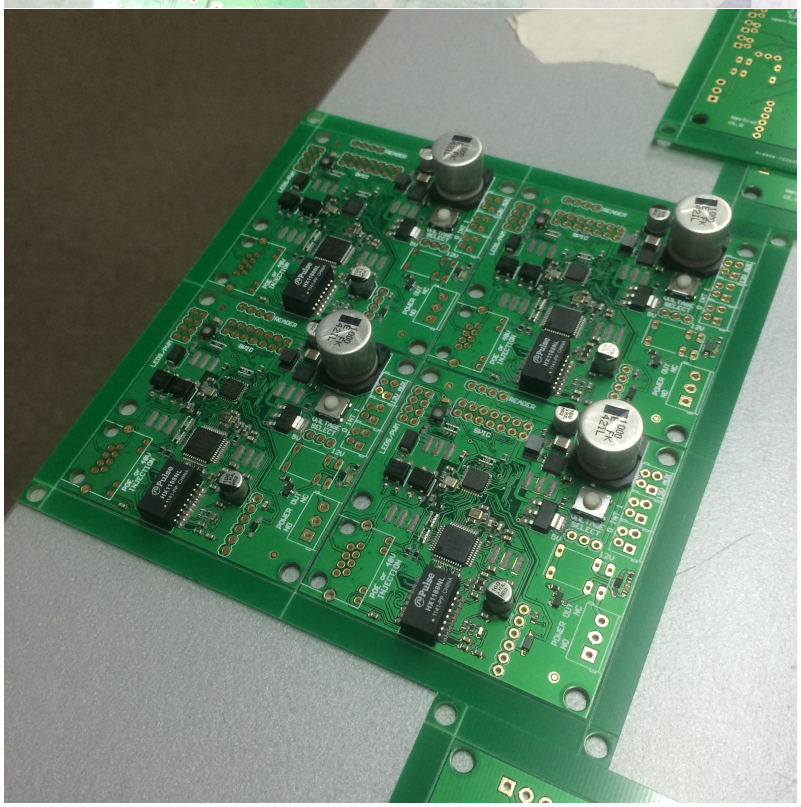
Block diagram



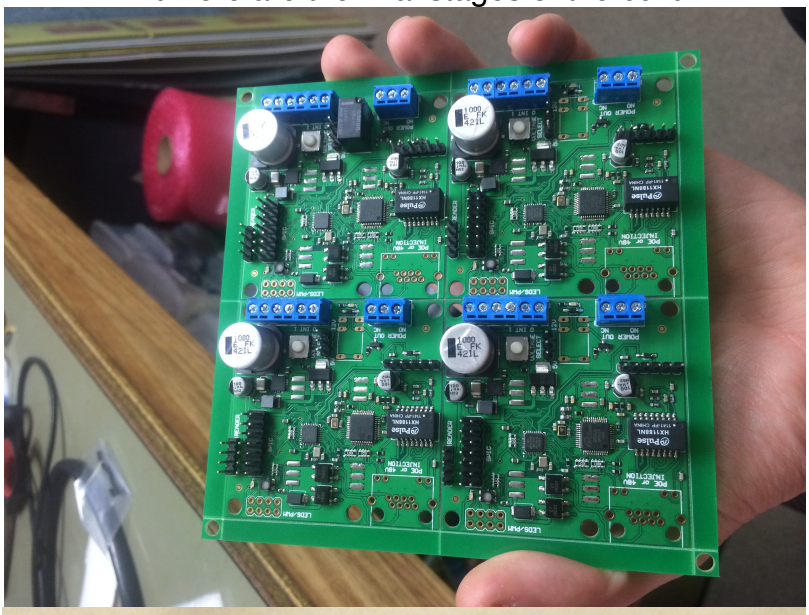
Project Photo’s

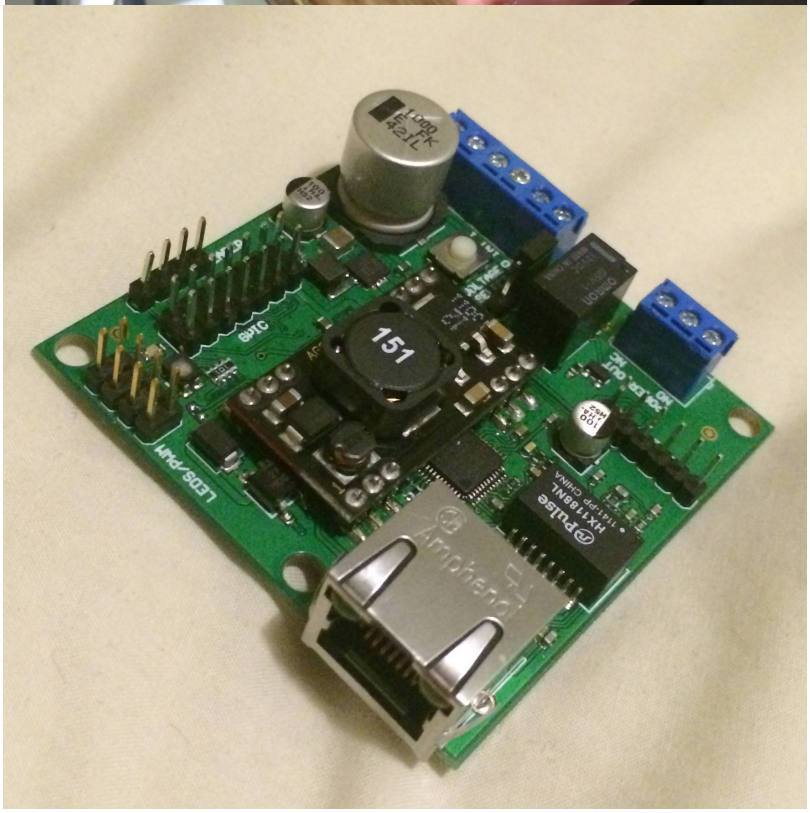
These are some of my build photo’s, I had a solder mask made from Mylar sheet that was laser cut. The two photo’s on this page shows the placement partly and fully placed ready for reflow.





And here are the final stages of the build.





Code Snippet

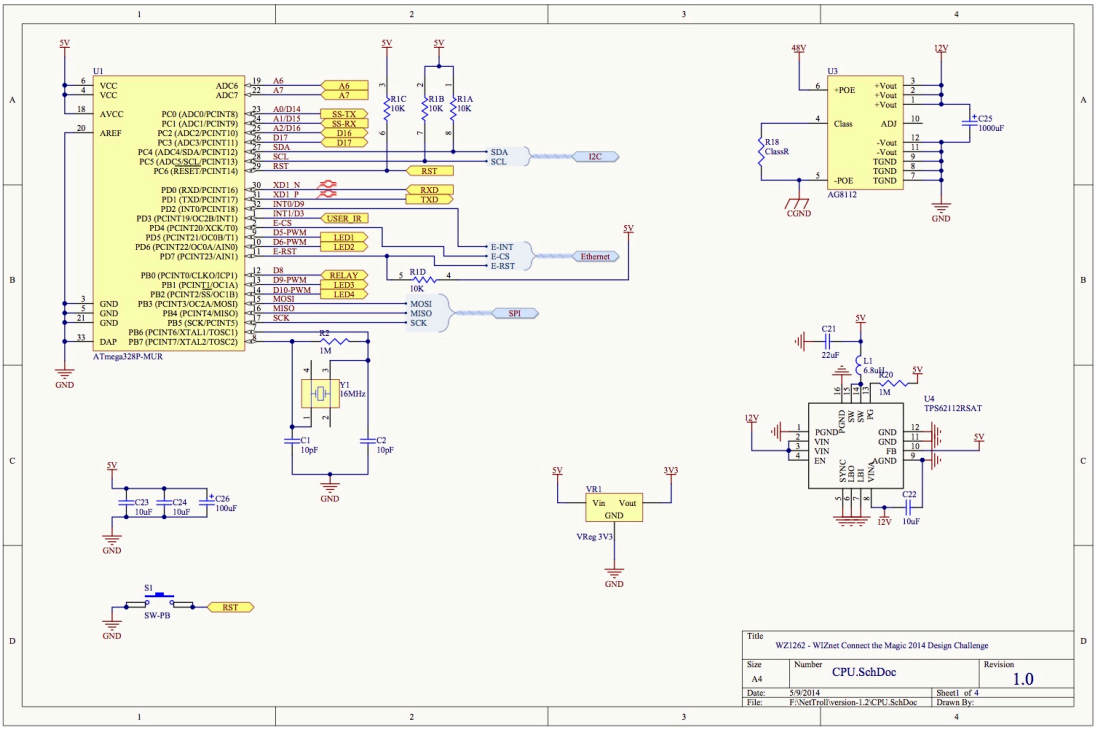
The code below is the main loop from an Arduino 1.5 project file; the main logic of this loop does the following.

- Blink the yellow led at a frequency of 2 seconds as a heartbeat.
- Check to see if we should enter program mode
- Has a RFID card been scanned
    - Check the local EEPROM for the card
    - If found, open the door and send a log to the server
    - If not found ask the server for permission, open the door if allowed and record details to EEPROM for next time
- Check to ensure we still have Ethernet connectivity
- Lock the door when the time expires

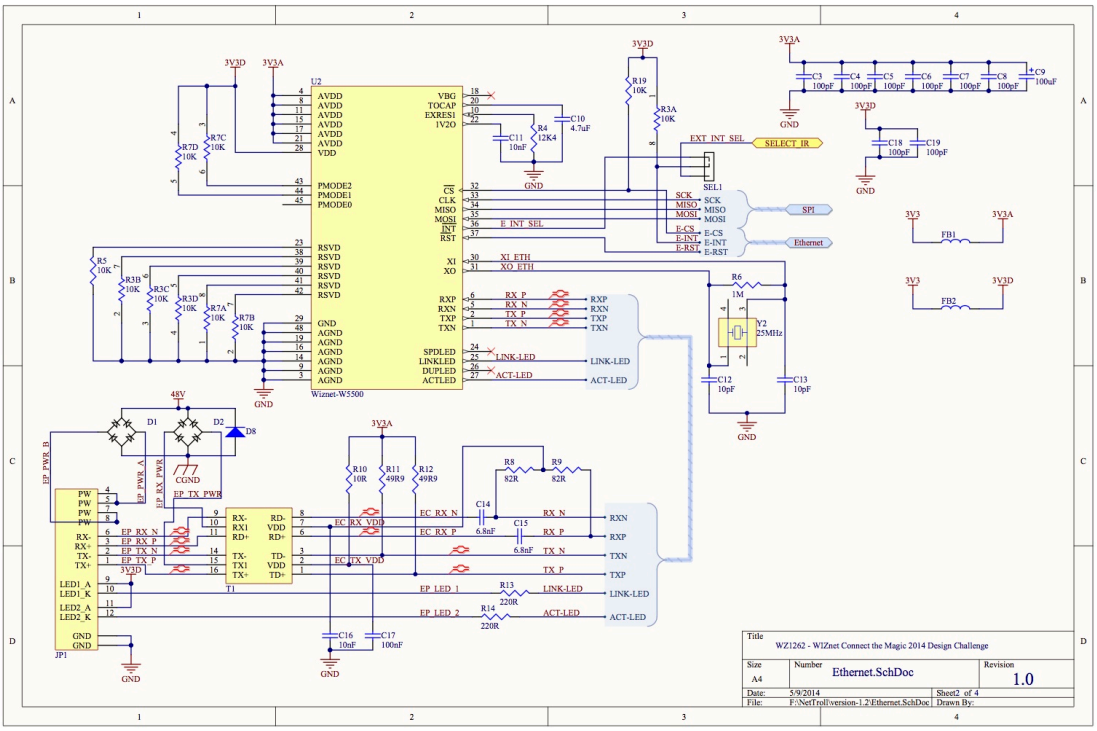
```
void loop()
{
   LEDS.toggle(LEDS_YELLOW, 2000);
   MENU.check();

   if(RFID.read(&rfidTag))
   {
      MEMORY.getNetworkInfo(&mySettings);
      Serial.print(F("RFID Tag:"));
      Serial.println(rfidTag);
      LEDS.off(LEDS_YELLOW);

      if(MEMORY.accessAllowed(&rfidTag)) // is tag in local EEPROM?
      {
         LEDS.on(LEDS_GREEN);
         // open door for 2 seconds and log to HTTP remote
         DOOR.unlockDoor(2000, &rfidTag, &mySettings.id, mySettings.deviceName);
         LEDS.off(LEDS_GREEN);
      }
      else if (ETHERNET.check_tag(&rfidTag, &mySettings.id, mySettings.deviceName) > 0)
      // unknown key, check what remote server has to say ( server logs it) ?
      {
          LEDS.on(LEDS_GREEN | LEDS_RED);
          DOOR.unlockDoor(2000); // open door for 2 seconds , no logging

          // Record Card for next time
          RFID_info newCard = {rfidTag, now()+SECS_PER_WEEK};
          MEMORY.storeAccess(&newCard);
          LEDS.off(LEDS_GREEN | LEDS_RED);
      }
      else
      {
         LEDS.blink(LEDS_RED);
      }
   }
   ETHERNET.listen();   // local http server handler.

   NETWORKCHECKER.listen();
   DOOR.locktimeout();

}
```

Schematic’s

CPU and Power Converters (48v -> 12v -> 5v -> 3v3)



Ethernet, Magnetics and 48v power rail coming in



Extra Information

Pin outs

All the unused or reusable pins have been placed on one large header for easy use. This allows for a ribbon connector or individual device connectors side by side in the one location keeping wiring to a minimum.

The pin out of this has been laid out to be compatible with off the shelf modules; this allows great flexibility when placing a design together.

The I2C port has been laid out to be the same as the RasperyPi, this allows modules like the Real Time Clock (RTC) module available from hobby stores like Seeedstudio.

Any RFID reader that supports UART, SPI or I2C can be used, the code has been written to use a UART connection. This hardware has been extensively tested and proven reliable using the 125Hz Frequency Readers also available from hobby stores.

Code Layout

When programing the code, I have tried to keep the code modular with a single configuration file; this allows easy expansion for other readers and devices. For example support for a LCD screen could easily be added if the system designer would like a message popup to the user when they swipe their card.

All the drivers including the Ethernet chip have been included in the project, this allows the code to be easily downloaded and compiled with minimal configuration. By doing this the drivers can be modified to use the same configuration settings file and becomes easier for the system designer to get code compiling for the device.