[Description 1](#_Toc796842647)

[V0.9 2](#_Toc130687972)

[Power circuit: 2](#_Toc830542252)

[Pumps control: 3](#_Toc1581074070)

[Motor Control: 3](#_Toc1865145696)

[Solenoid valves control: 4](#_Toc1846084463)

[User interface: 5](#_Toc611917481)

[Heating element: 6](#_Toc705486995)

[Sensors: 7](#_Toc1451161667)

[Buzzer: 8](#_Toc1288645310)

[Communication Ports: 8](#_Toc368371457)

[Microcontroller pinout: 9](#_Toc1679076138)

[V0.92 (in machine after ) 10](#_Toc973718169)

[Power circuit: 11](#_Toc1384057451)

[Pumps control: 12](#_Toc1770268440)

[Motor control: 12](#_Toc1342332444)

[Solenoid valves control: 13](#_Toc441690670)

[User interface 13](#_Toc1074160076)

[Heating element: 14](#_Toc1448263776)

[Arduino USB Host shield dock: 15](#_Toc72465910)

[Sensors: 16](#_Toc1381948264)

[Buzzer: 16](#_Toc1861782406)

[Communication Ports: 16](#_Toc1490467279)

[Microcontroller pinout: 16](#_Toc702601721)

[List of features in planning: 18](#_Toc10519304)

Description

The CP800 circuit is designed to fulfil the function of developing films with drum rolling and it consists of the following requirements

1. Control the fluid flow of developing film
2. Control the temperature of the fluid
3. Control the motor for the rolling action
4. User interface

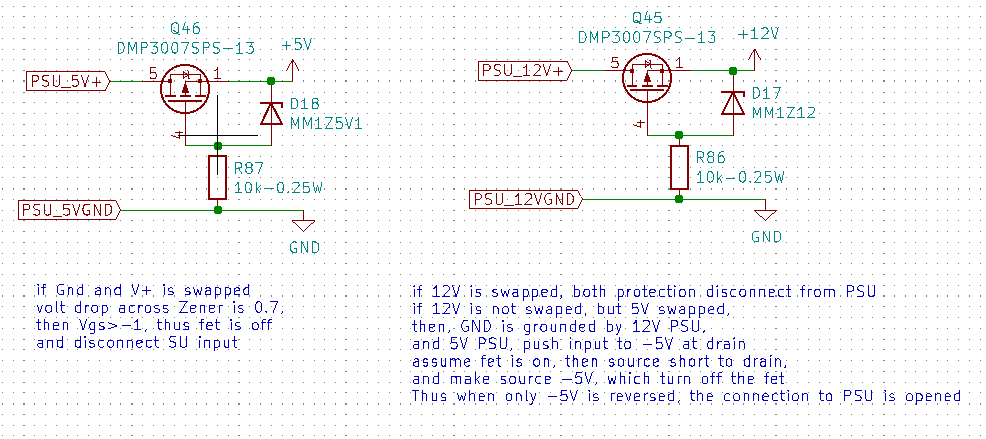
# V0.9

The original working prototype for CP800

The function of the board is broken down into the following sections:

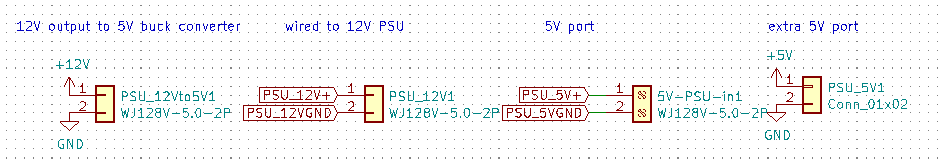
1. The power transfer to each of the components below
2. The pumps
3. The motor
4. The fluid sensors feedback
5. The solenoid valves for flow control
6. The user interface
7. The heating element

## Power circuit:



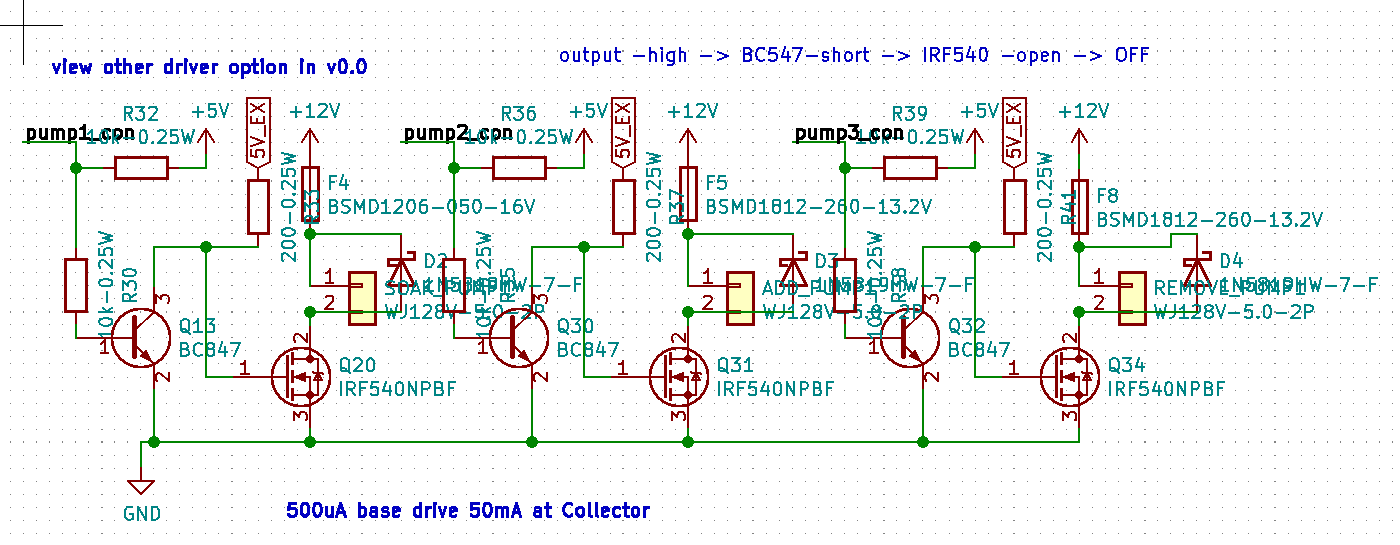
Both 12V and 5V comes from the external switch converter and are reverse polarity protected with low on resistance p-channel MOSFET and Zener diode.

The Arduino is powered via the 5V pin bypassing the LDO that required a higher voltage to power. And all 12V and 5V components will be supplied by these two voltage sources.



There are four power connectors, one is the input from the 12V PSU, one is the 12V output after the reverse polarity protection, one is 5V voltage input from the buck converter, and the last one is an extra header for a 5V reference.

## Pumps control:



There are three output ports for pumps. They are 2pin output.

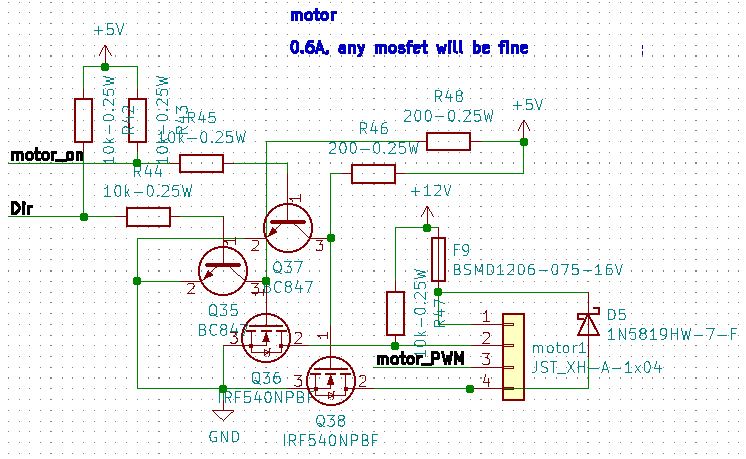
Each pump is driven by a power MOSFET to switch them on and off with a low side BJT transistor as the gate driver.

A high output in the microcontroller will allow gate to discharge thru the BJT quickly, and achieve a faster turn off of the pumps. Meanwhile, a low output on the microcontroller will turn the BJT off, and the MOSFET gate is charged thru the low resistance pull up resistor, to achieve a regulated turn on time.

The pull up resistor in the microcontroller signal pin is implemented in the case where these pins are not programmed. The pull up resistors will ensure the pump is off when these pins are not programmed.

Fuses are installed to prevent over current, and a flyback Schottky diode is used to limit voltage surge by the induction load from the pumps.

## Motor Control:



The motor control is very similar to the pump control as they are both inductions loading presumably.

The motor we are using has 5 pins control: PWR, DIR, PWN, FB, GND. We don’t use the current feedback wire, so it is left out of the port output intentionally.

The motor is then connected to the 4 pins JST XH output on the board.

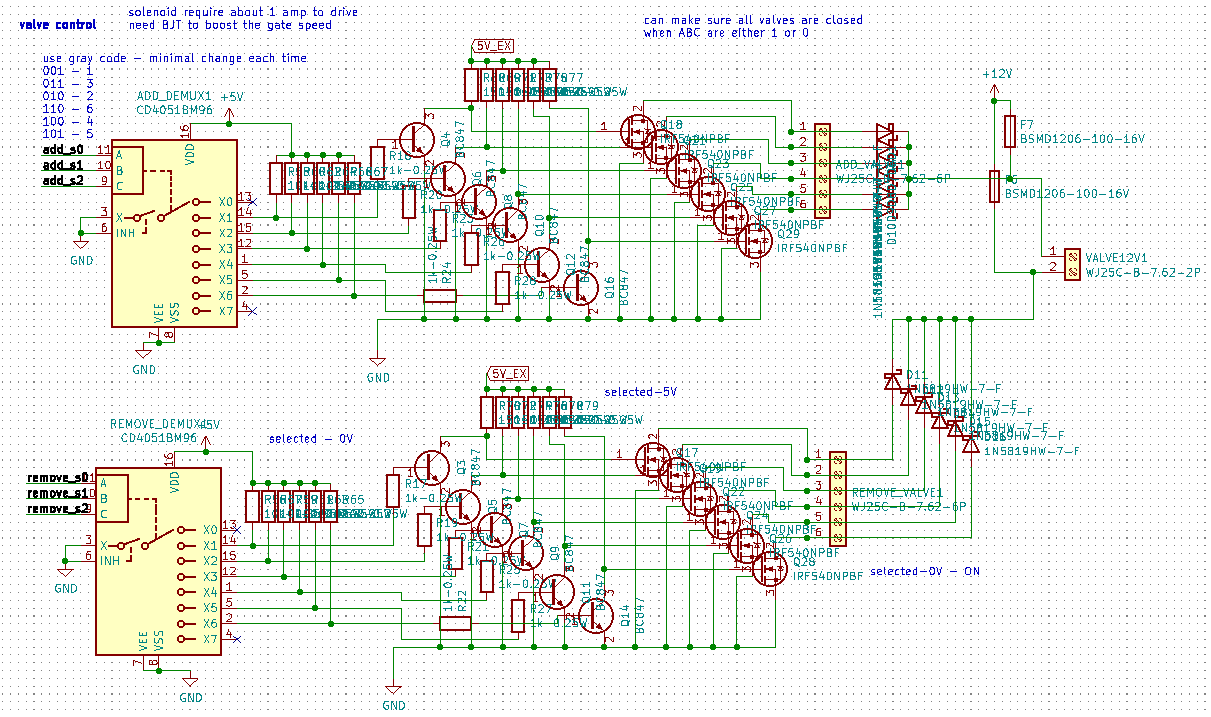
In order to turn on the motor, we need the PWE pin to be 12V, and the GND pin to be 0V. There is a fuse of the appropriate current to limit over current, and there is a flyback Schottky diode to limit voltage surge.

Both DIR and GND pins are controlled by power MOSFETs to switch between 12V and 0V. A pulled up resistor of 12V is in place for the DIR pin, so there is a default DIR of rotation for the motor, if the pin is left hanging.

The low side BJT allows a quick discharge of gate, and the low resistance pull up allows a fixed time fast charging up to the gate.

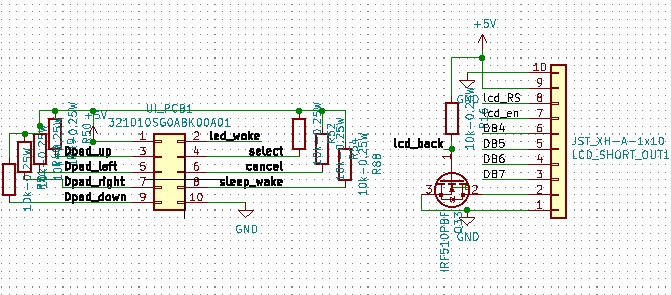
The pulled up in the microcontroller pins are to make sure the DIR and 0V pin are forced to 12V and keep the motor off, if these pins are not programmed by the microcontroller.

## Solenoid valves control:

Both add and remove valves are controlled in the same way. Since solenoid is a highly inductive load, we have to apply a voltage of 12V across each pair of contact on the solenoid block. Similar to motor and pumps, a fuse to protect from over current and flyback Schottky diode for voltage surge.

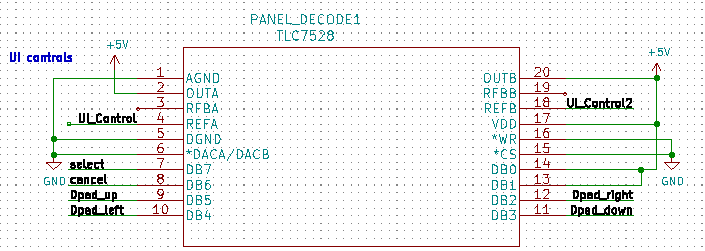
Each solenoid is switched on and off by a MOSFET with a low side BJT driver for fast drain in gate and a pulled-up resistor in case of unprogrammed microcontroller pins to the demultiplexer, CD4051.

## User interface:



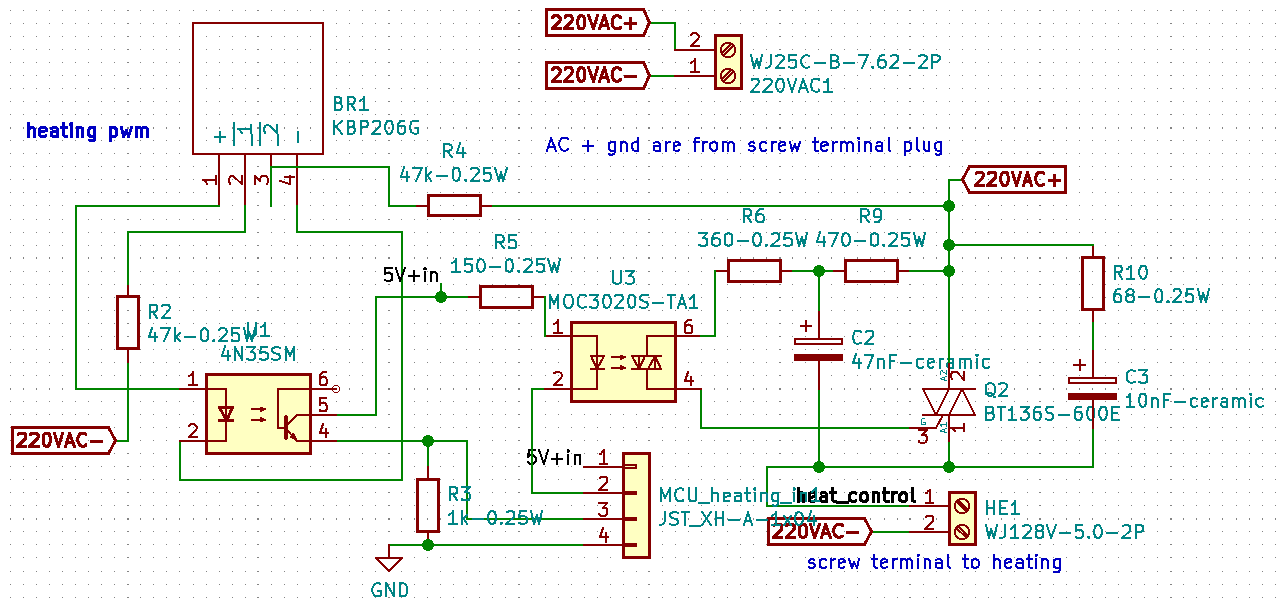
There are two connectors between the motherboard and the Dpad board. These control the user interface. The 10ways IDC connector is the connection with each button and LED on the Dpad. The 10ways JST XH header is the signal going to the LCD screen.

The screen is turned on and off by MOSFET, the program is currently locked to stay on. Future programming is required to control the backlight.



The signal from the Dpad will be processed by a DAC with a R-2R structure, to convert the 6 inputs from the Dpad to a binary coded 8bit voltage. Note that only UIcontrol analog signal is used, the second output is left connected. Now.

## Heating element:



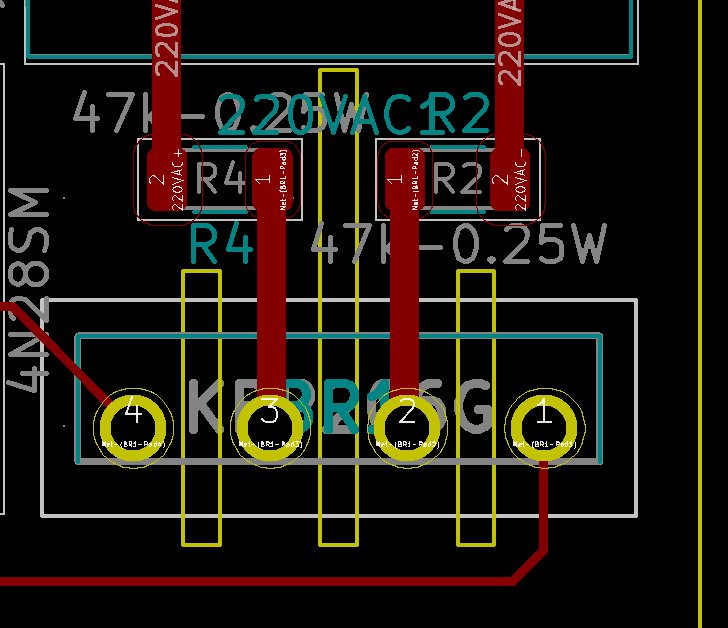
The heating element chosen is one that is powered by AC 240V input, at the connector, HE1.

KBP206G is the rectifier to convert AD into DC voltage.

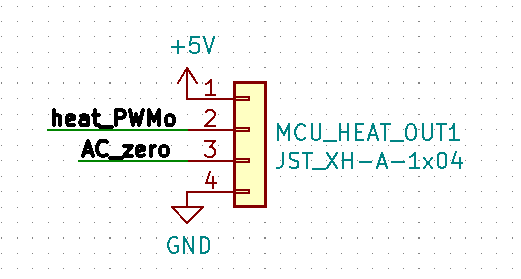
The 4N35SM photocoupler will convert the rectified wave and generate an on and off 5V signal to the microcontroller to locate the zero-crossing point of the input wave. This will allow the microcontroller to align the output from this zero-crossing moment if necessary.

MOC3020 is the photocoupler, between DC to AC using a triac. This component allows the microcontroller to sink the current at the cathode (the negative, pin 2) of the MOC3020 diode. Allowing the microcontroller to control the pulse powering the heating element by driving the BT1365 triac.

R6, R9 and C2, are there to protect the MOC3020 from load, and R10 and C3 form the snubber to protect the triac.

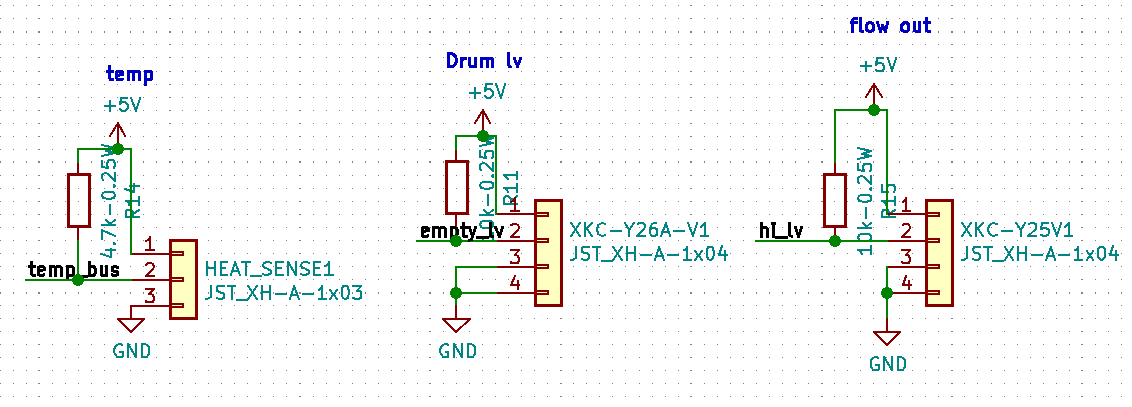


Additional cutout on PCB to increase the effective isolation between the 240V on each contact of the rectifier.



The AC circuit is located at a different part of the PCB, and the pinout for controlling the heating element is done by a 4ways connector as shown above. A jumper cable will be required to connect this connector, MUC\_HEAT\_OUT1, with the one input 4ways connector, MCU\_heating\_IN1, connection the MOC3020 and 4N35 IC.

## Sensors:



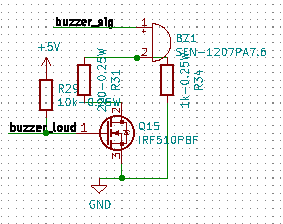
There are three main sensors: temperature, tank high level, and flow out sensor

Temperature sensor is currently unuse, it is supposed to serve as the feedback to control the heating element. It is using a one-wire bus to transmit its digital data. The sensor used here is DS18B20.

The tank high level sensor is a 4 pins socket connecting to a capacitance sensor mounted to the tank and will trigger a on signal when the water level is raised.

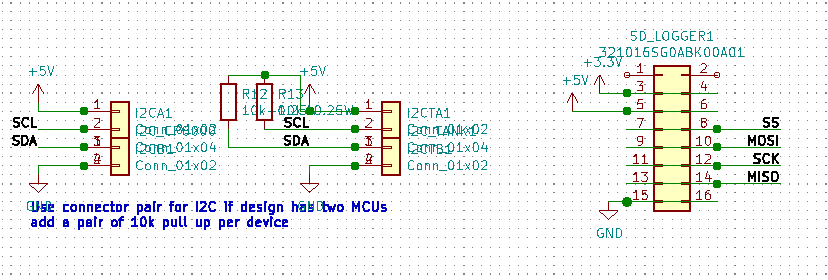
The flow out sensor is a 4 pins socket connecting to two capacitance sensors clipped to the outlet tubbing from the tank to the solenoid valve, and will trigger a signal when they sense the tube has fluid inside.

## Buzzer:



A buzzer is integrated to the board, this allows the program to give out audio clues to the user/operator when things happen. See software documentation to see the list of audio clues.

## Communication Ports:

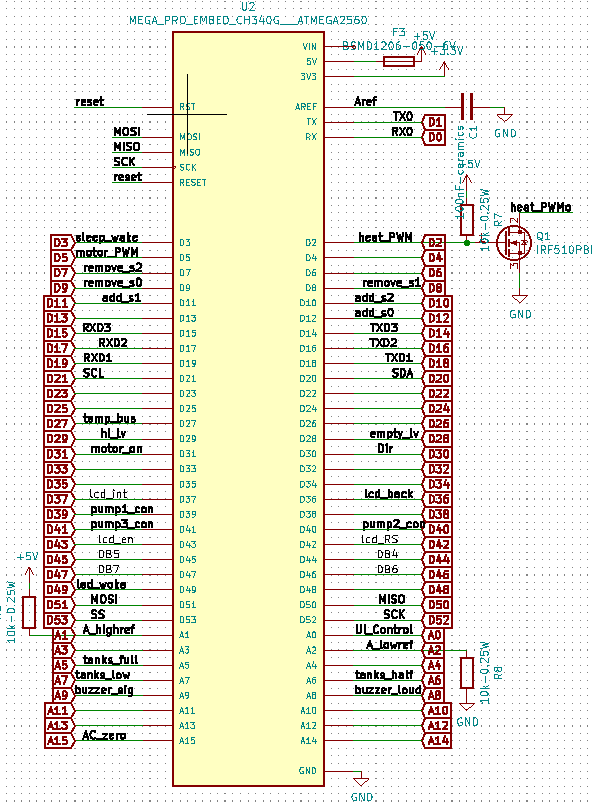


There are two types of communication port accessible from the boards via three ports:

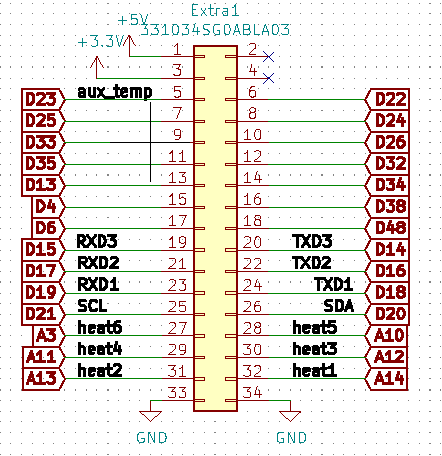
There are two I2C ports, 4 ways, with one of the port has an extra set of pulled up resistor for a new additional device

There is a 16way IDC SPI communication port. This is currently used for the SD card module. The hardware is suitable for any SPI device. The parallel pin socket allow another device to be piggy-back to the same port.

## Microcontroller pinout:



A pinout diagram of the microcontroller to all the other components on the board. Some basic components such as transistor and resistor are added depending on the signal the pin is interfacing with. This would change depending on the programs being uploaded to the microcontroller.



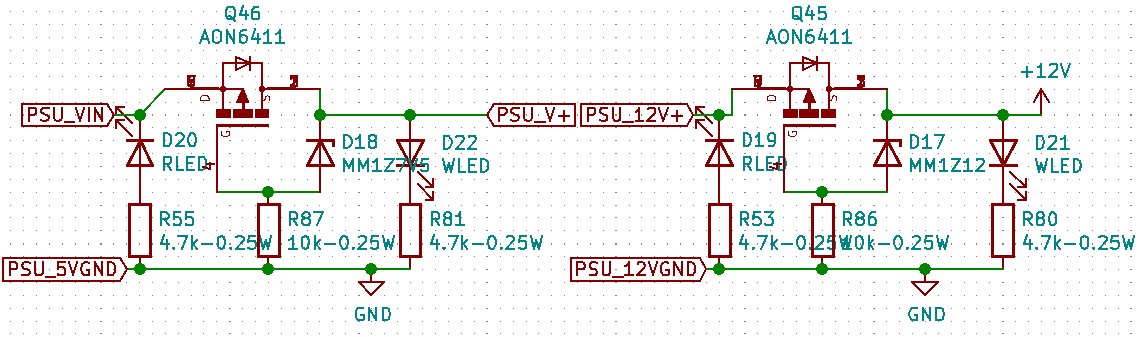
All the extra pins are tentatively grouped to an IDC header for future development and expansion.

# V0.92 (in machine after )

Changes:

1. Removed pull up resistor for direction pin of the motor
2. Number of extra pins reduced
3. One additional IDC socket for joystick shield
4. Pin out header socket for USB host shield, required one additional stacking pins
5. Removed the extra ICSP pins, it is now used by the USB host shield
6. Use one 5V LDO (AMS1117) onboard required low voltage input of at least 6.5V
7. Arduino mage gets power from Vin around 6.5V and signals powered from onboard 5V LDO (AMS1117)
8. One PISO shift registers for Dpad buttons input are available instead of the R-2R ladder analog output to buttons
9. Red onboard LED for reverse voltage input for both 12V and low voltage terminals
10. Green onboard LED for forward voltage input for both 12V and low voltage terminals
11. Mounting holes increase from M3 to M4, holes position remains the same
12. Size of board increase in the X-axis

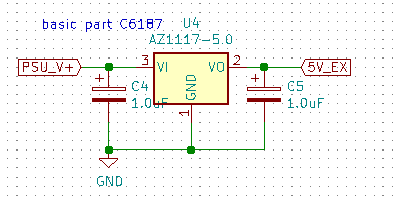
## Power circuit:



Both 12V and low voltage (6.5V) from the external switch converter are reverse polarity protected with low on resistance p-channel MOSFET and Zener diode.

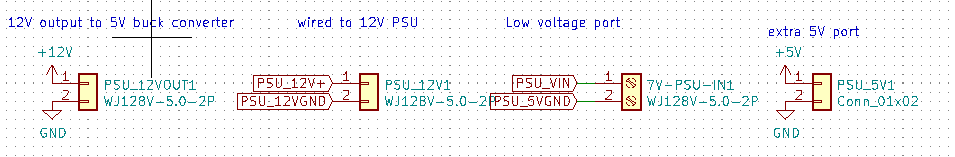
When reverse polarity occurred, the pMOSFET will shut down the main circuit from the high voltage side, but the red SMD LED will turn on as an indication for the user that they had reversed the input leads

When the correct polarity is inserted, the pMOSFET will shunt the high voltage side leads, and activate the green SMD LED.



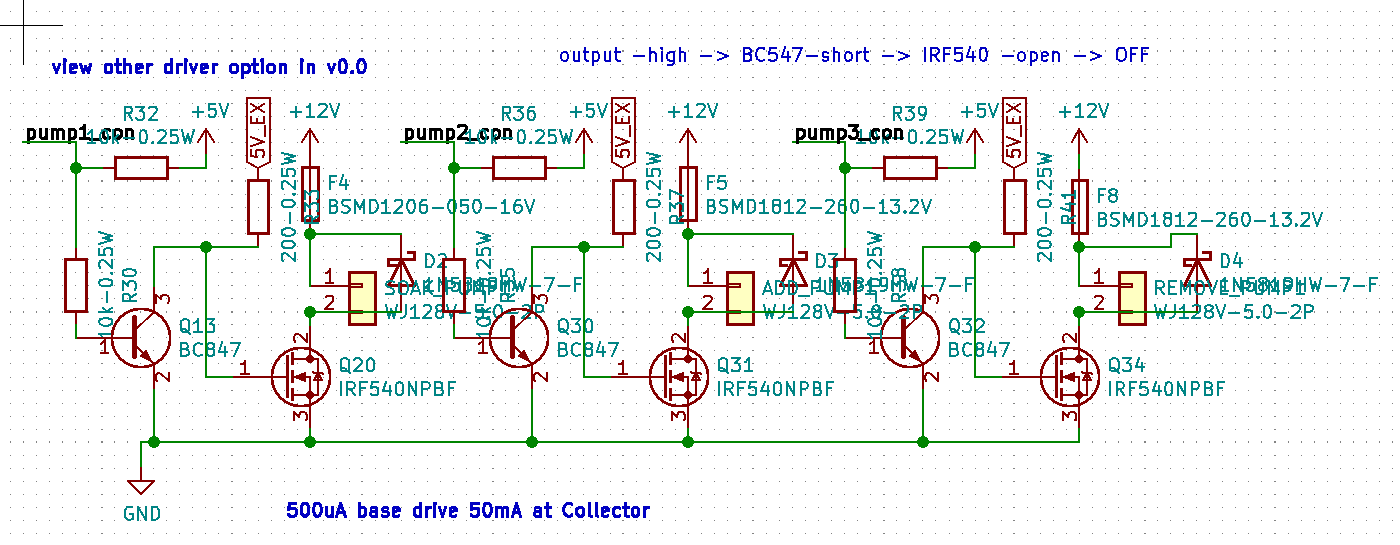
One additional 5V regulator is implemented to share the loads required by the 5V sources. This is the same chip as the 5V LDO on the Arduino mega pro mini

In the schematics diagram, 5V is the one powered by the Arduino, while all 5V\_EX connection is powered by this additional chip.

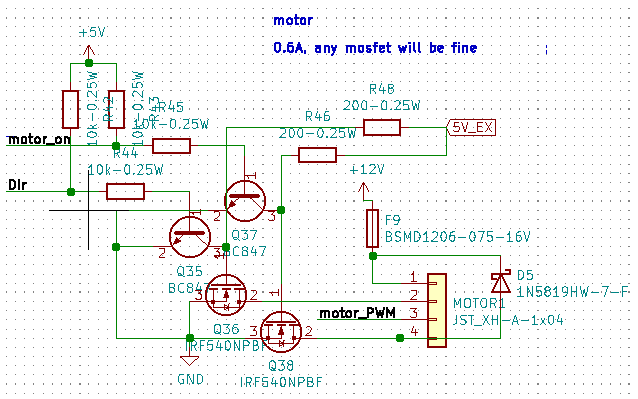


There are four power connectors, one is the input from the 12V PSU, one is the 12V output after the reverse polarity protection, one is low voltage input from the buck converter, and the last one is an extra header for a 5V reference.

## Pumps control:

Remain unchanged

## Motor control:



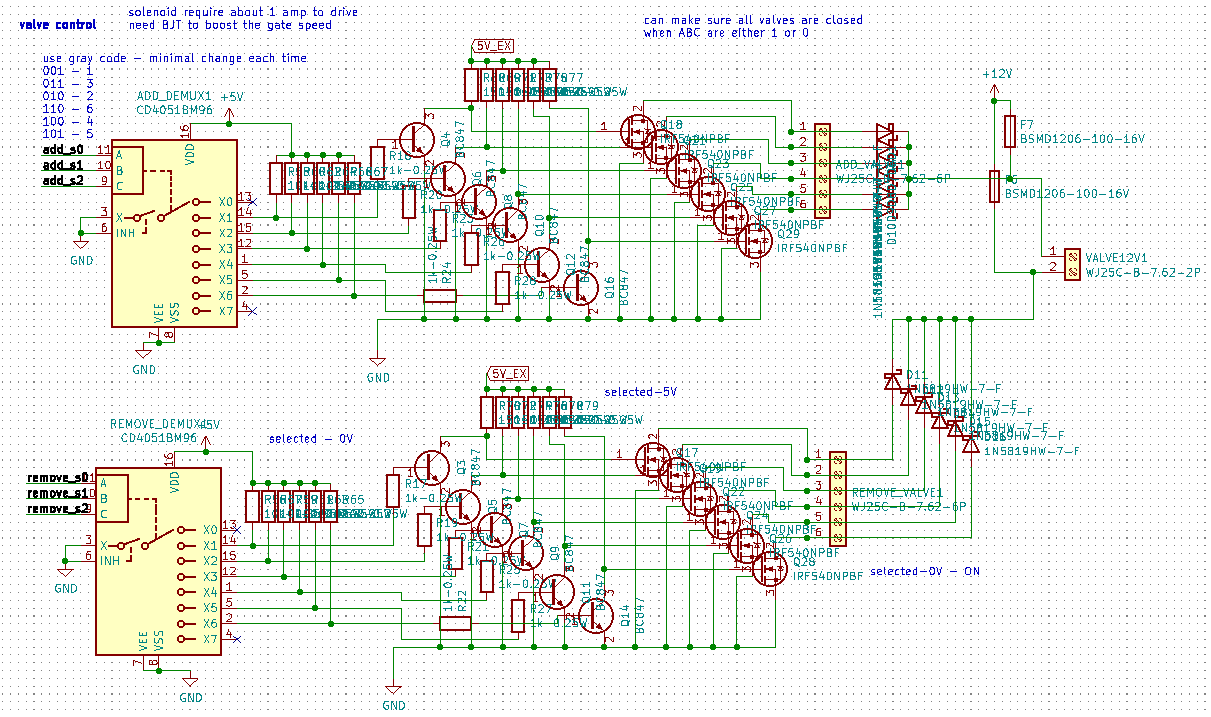
The only changes made are the direction pin of the motor. Now the pin is left floating/disconnected if it is not shorted to GND for backward direction

The change is made because the motor we used is not a raw motor, it is a DIR and PWM controlled motor. Therefore, there is a controller board inside the motor assembly. Depending on the quality of the motor, if there is a current path inside the motor assembly, the additional pull up resistor outside the assembly will affect the voltage level on that pin.

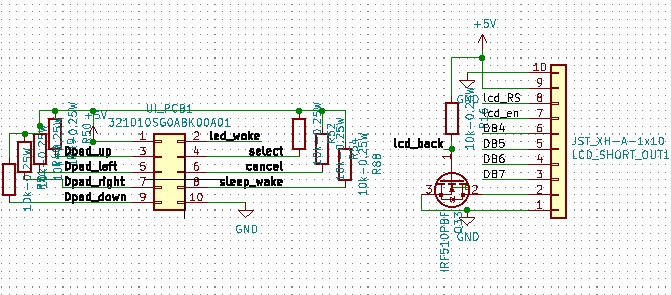
Also, according to the product datasheet, the DIR pin could be left disconnected for forward direction, thus proving there is an internal pull up inside the motor assembly.

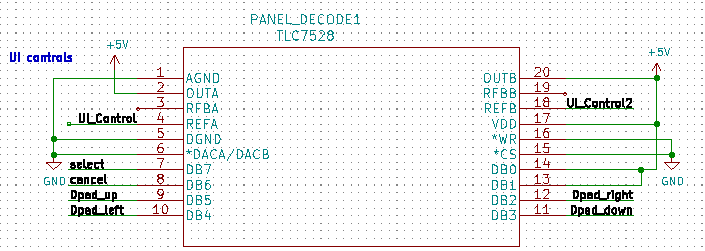
The rest is the same as the previous version.

## Solenoid valves control:

Same as previously

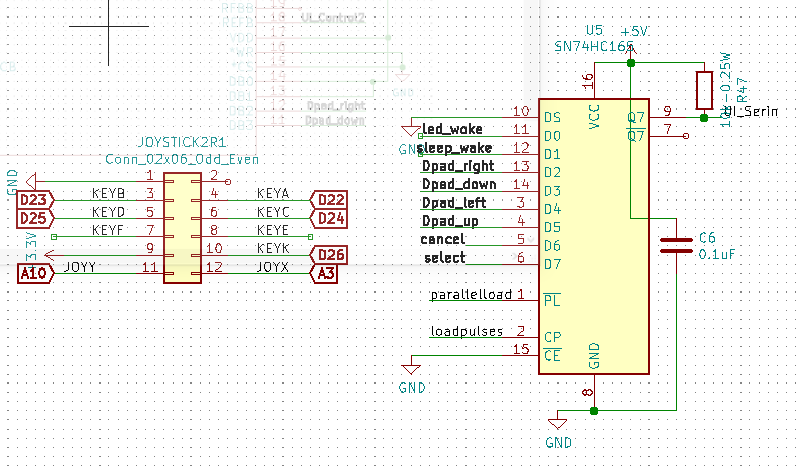
## User interface





The Dpad function works as previously

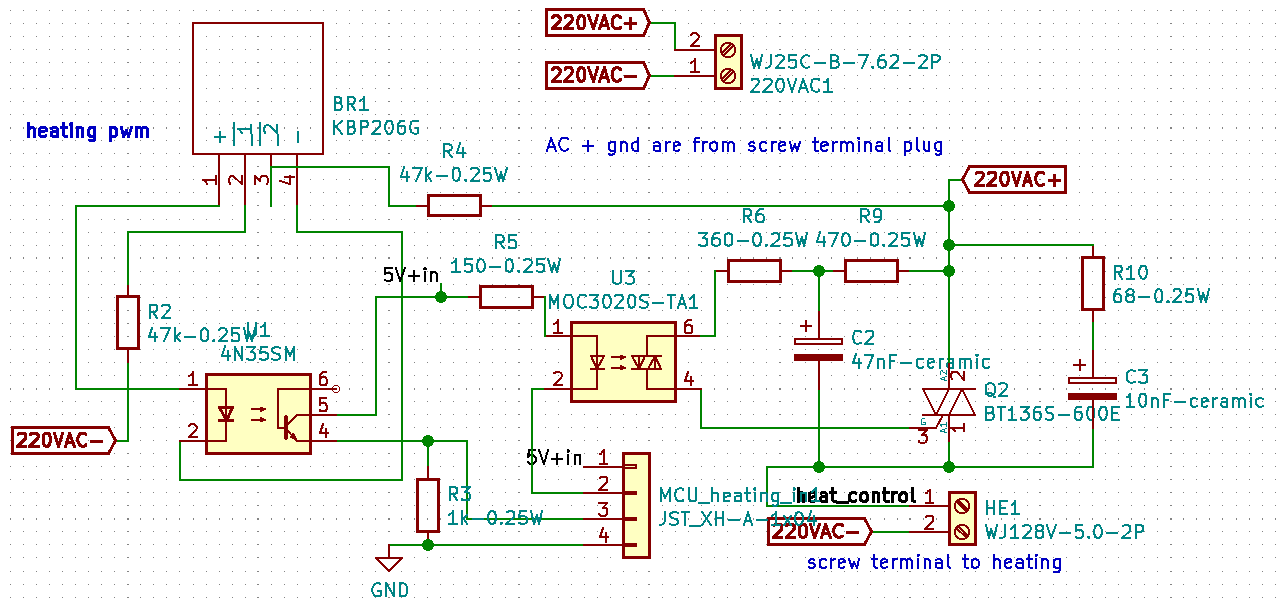
In v0.92, there are two additional components added for the user interface



Firstly, the 12ways IDC connector allows the microcontroller to interface with Arduino joystick shield. Since we don’t plan to use key E and F, they are left disconnected with the microcontroller here. This is a optional addon, the Dpad functions as usual without the connection with joystick shield, and the joystick shield will work simultaneously with the Dpad.

The addition of a parallel in and series out shift register allows pure digital signal to be transmitted to the microcontroller for all the Dpad buttons. This required additional programming that allowed the signal to be read via the UI\_Serin input for all 8 buttons.

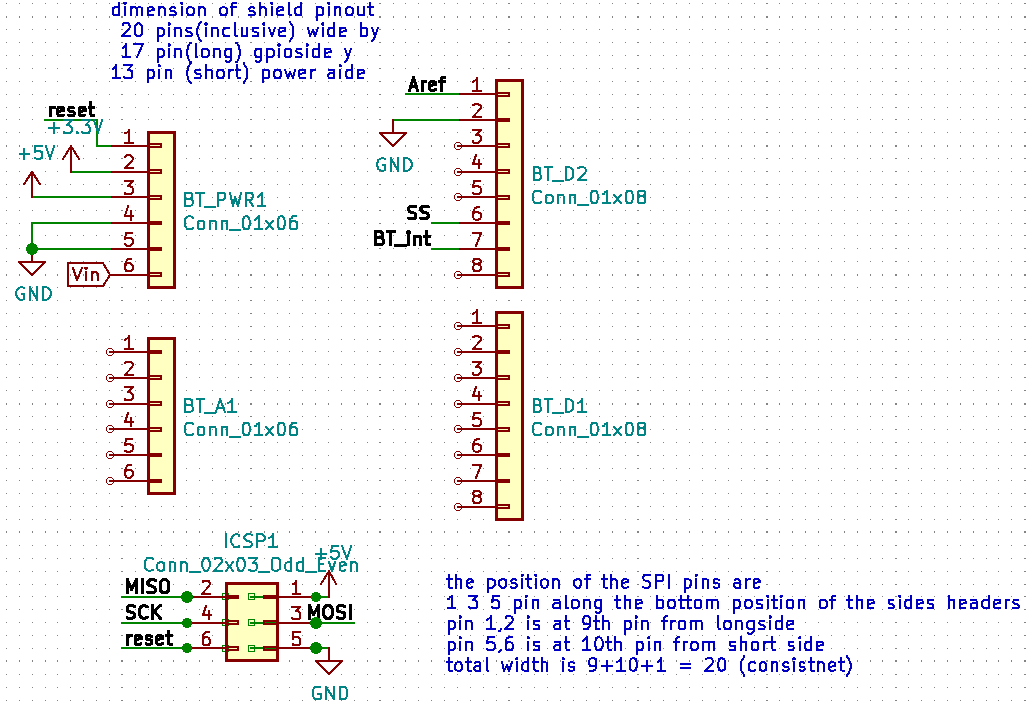
## Heating element:



Same as previously

But, DO note that MOC3020, and 4Nm5 might need to be changed to other equivalent components due to availability.

## Arduino USB Host shield dock:

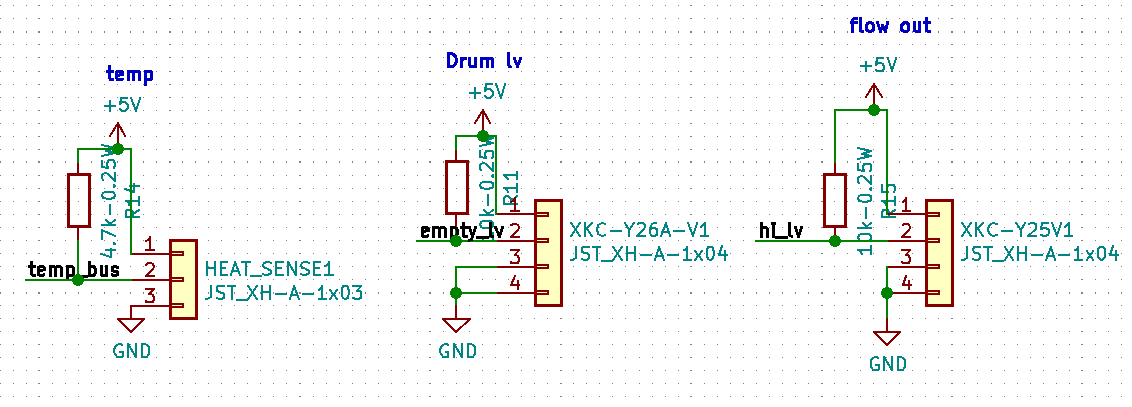


This layout allows the Arduino USB Host shield to be connected simply by docking it over these docking headers. Most of the bypassing digital and analog pins from the shield are unuse. They are included for structural reasons.

The USB host shield converts the USB signal into SPI interface to the microcontroller.

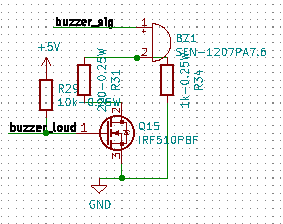
This has been tested and used with a Bluetooth USB dongle, and communicates with a PS5 controller to act as the user interface to navigate the menu.

## Sensors:



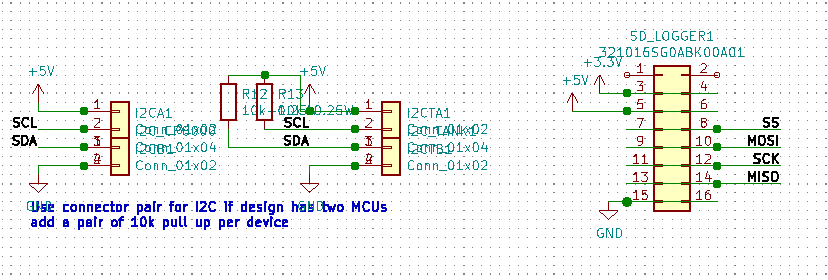
Same as previously

## Buzzer:



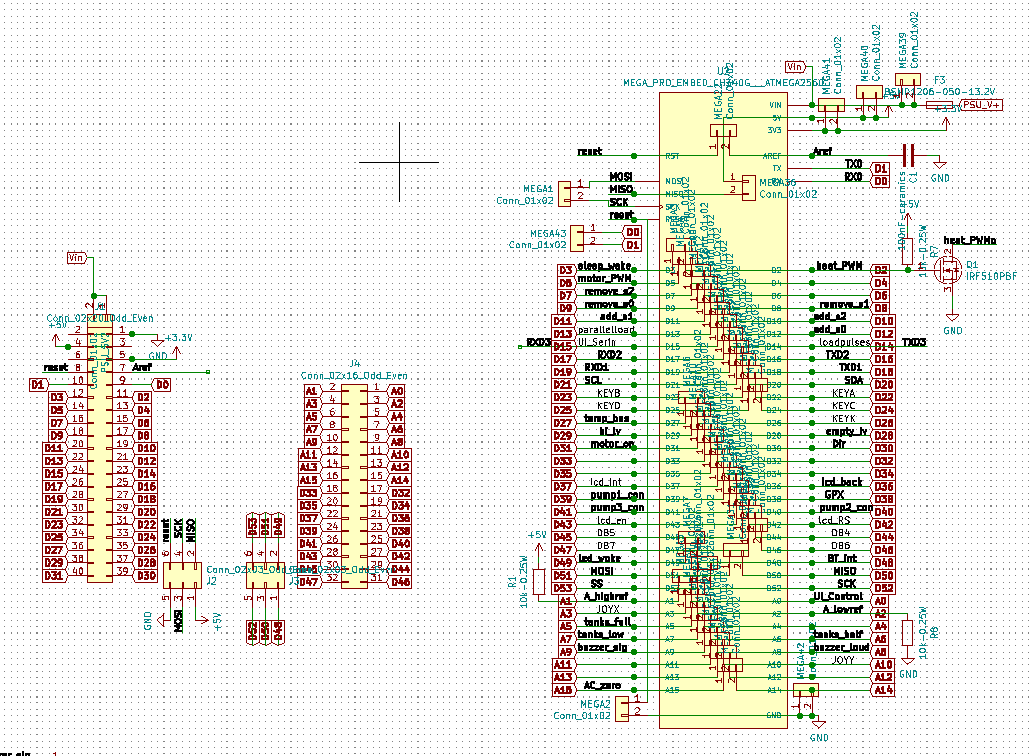
Same as previously

## Communication Ports:

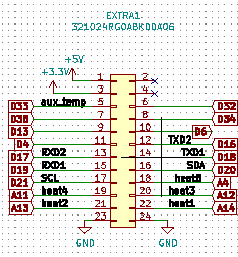


Same as previously

## Microcontroller pinout:

The left side of the picture is the header arrangement on the board for docking the arduino board. The right is the labeled pinout of the Arduino.

A additional fuse is added to the Vin to the Arduino board, since in v0.9, power is feed via the 5Vport of the Arduino bypassing the 5V LDO onboard.



Extra pin unused pin is group to an extra IDC header for future expansion. This has been significantly reduced due to the module expansion and component compatibility from v0.9 to v0.92.

The firmware is made backward compatible, so the program being uploaded to this board that processes all the added component to this iteration of PCB can also be uploaded to v0.9 without causing a bug. However, all new features in v0.92 will not reflect on v0.9 unless the corresponding extra pins are wired externally.

# List of features in planning:

Please read this in conjunction with the software plan, since back-end development of the software will be highly tied with the hardware development.

* Multiple heating element array control, if each storage chemical needs to be kept at a different temperature
* Temperature PID controller (software)
* Allow the use of 100VAC, there will be a transformer to step down higher AC voltage to 100VAC, this allows the machine to work in country with a different phase voltage, by converting them all to 100VAC.
* Program the microcontroller to control the backlight of the LCD screen if required. (software)
* Program that allows a selection to read from the digital input of the Dpad via the serial output from the shift register. (software)
* Implement an on and off software button using the LED button mounted on the Dpad. (software)
* Earth all solenoid to a common point.
* Overvoltage and undervoltage protection
* Quick self-diagnosis button or additional debugging display (led on communication...etc)
* Float sensor feedback for each storage tank