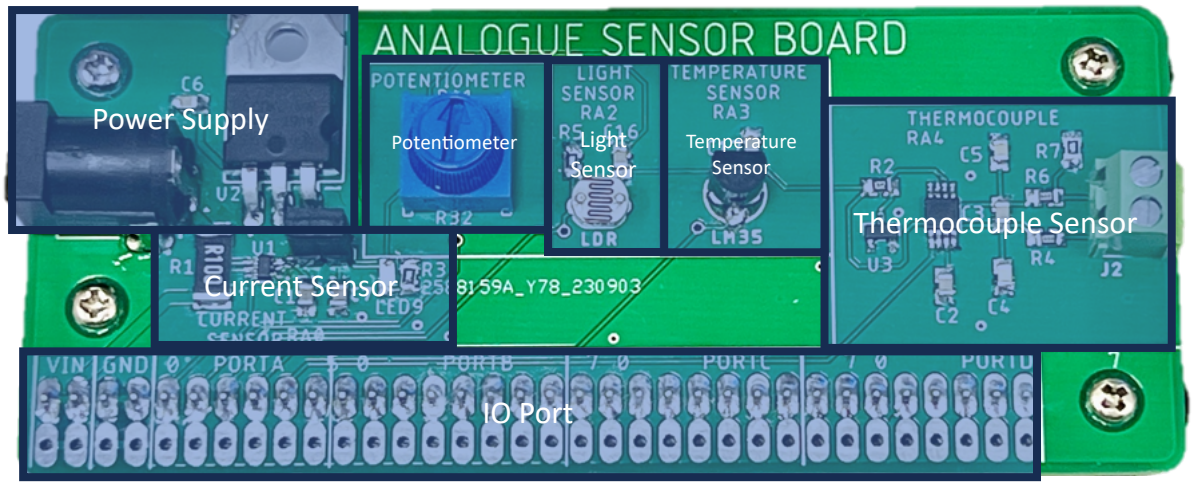
ANALOGUE SENSOR BOARD

The Machine Shop

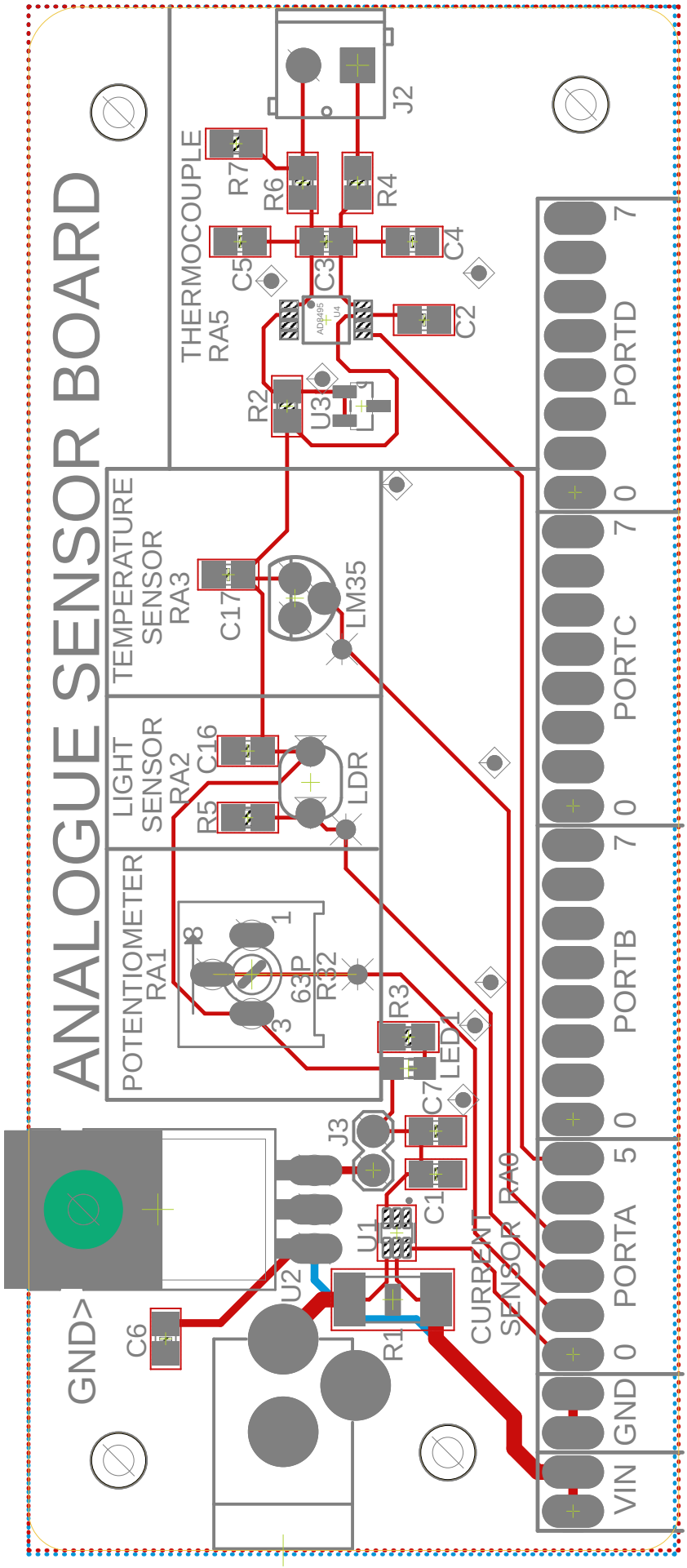
John Lawrence

Table of Contents

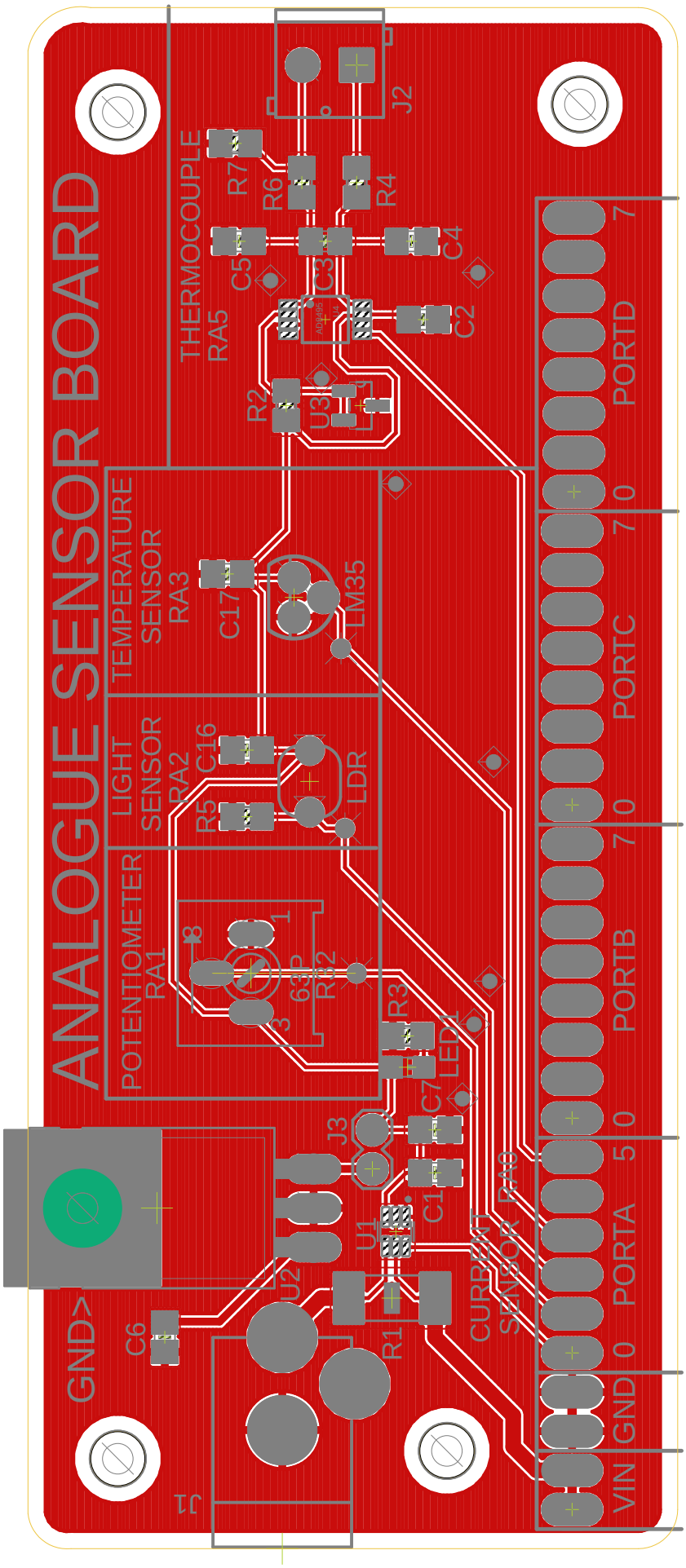
Layout



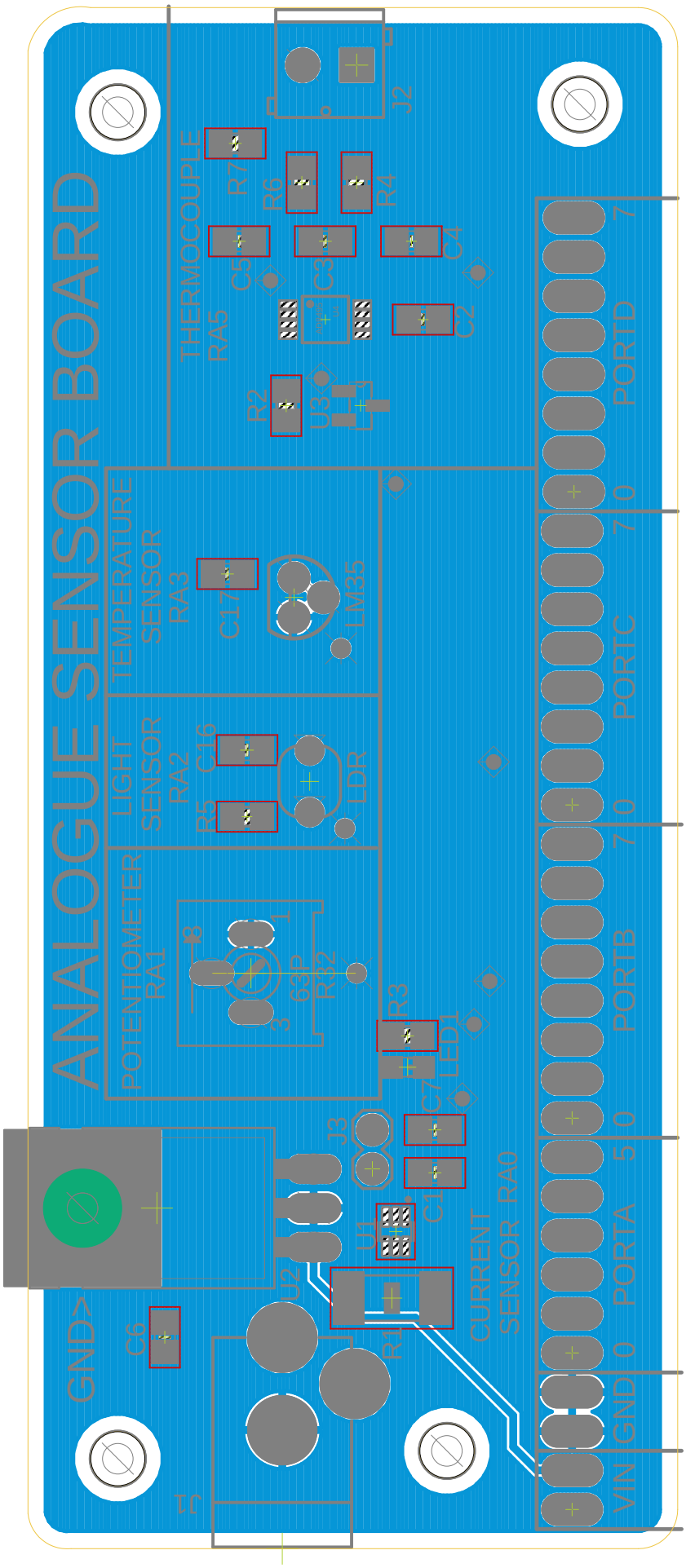
PCB Layout


ANALOGUE SENSOR BOARD
THERMOCOUPLE
RA5
TEMPERATURE
SENSOR
RA3
LIGHT
SENSOR
RA2
POTENTIOMETER
RA1
CURRENT
SENSOR
RA0
J2
R7
R6
R4
C5
C3
C4
C2
R2
U3
C17
LM35
C16
R5
LDR
R32
R3
LED1
C7
J3
C1
U1
U2
R1
C6
GND>
PORTD
PORTC
PORTB
PORTA
GND
VIN

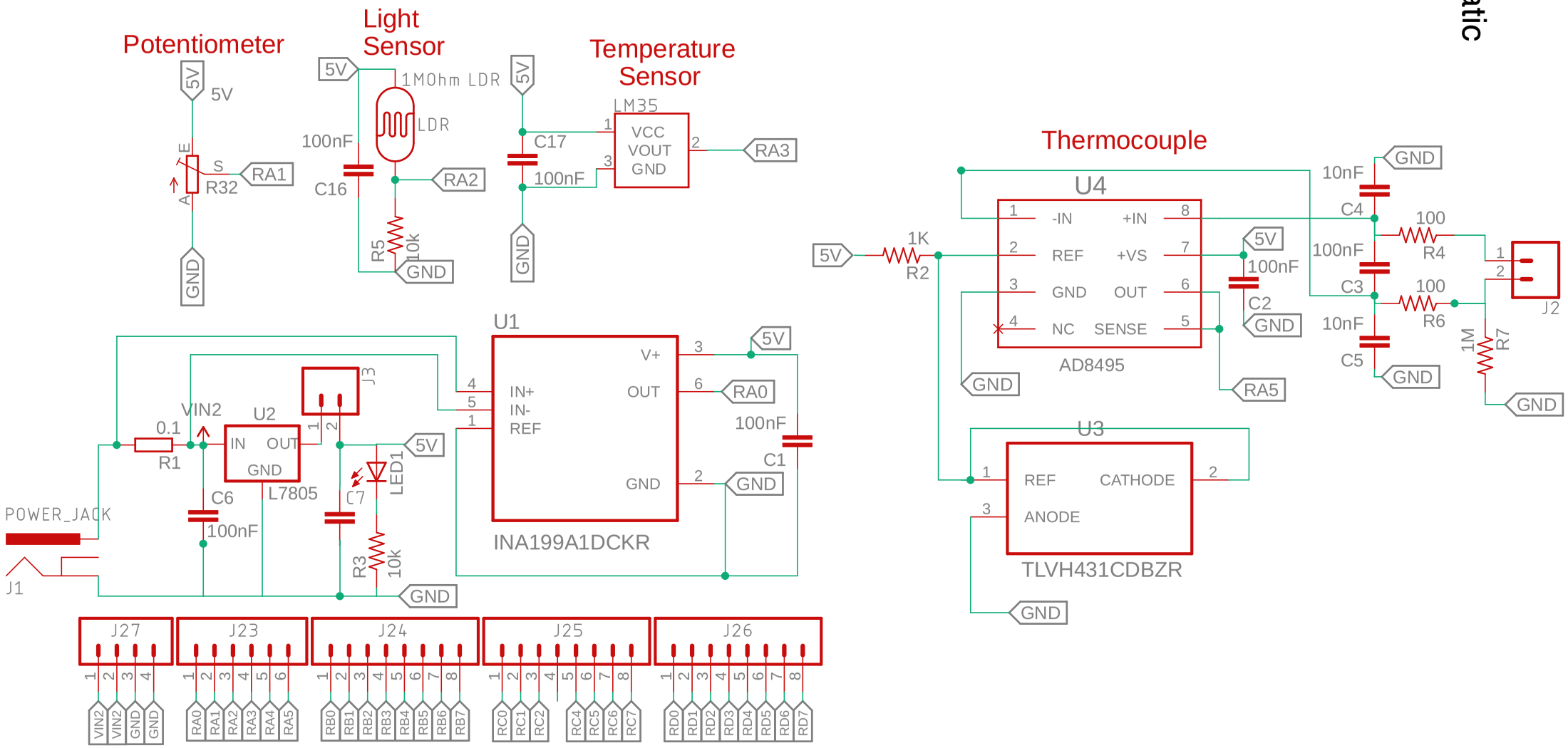
PCB Top Layer



PCB Bottom Layer



Schematic


Potentiometer
Light Sensor
Temperature Sensor
Thermocouple
5V
GND
R32
RA1
100nF
C16
1MOhm LDR
LDR
RA2
R5
10k
C17
LM35
VCC
VOUT
GND
RA3
U1
IN+
IN-
REF
V+
OUT
GND
RA0
100nF
C1
U4
-IN
+IN
REF
+VS
GND
OUT
NC
SENSE
AD8495
1K
R2
100nF
C2
RA5
10nF
C4
C3
C5
100
R4
R6
1M
R7
J2
U3
REF
CATHODE
ANODE
TLVH431CDBZR
VIN2
U2
IN
OUT
GND
L7805
0.1
R1
C6
100nF
J3
C7
LED1
R3
10k
POWER_JACK
J1
J27
J23
J24
J25
J26
VIN2
VIN2
GND
GND
RA0
RA1
RA2
RA3
RA4
RA5
RB0
RB1
RB2
RB3
RB4
RB5
RB6
RB7
RC0
RC1
RC2
RC4
RC5
RC6
RC7
RD0
RD1
RD2
RD3
RD4
RD5
RD6
RD7

PIC ADC Setup

To convert the analogue signals from the sensor board into digital form on the PIC microcontroller, The ADC must be setup correctly. The following code is for the PIC18F45K50 Microcontroller that is on the Modular PIC board. This can be placed at the start of the main loop or in a separate function which is called at the start of the main loop:

```
// Setup variable to store the data
float ADCvalue = 0.0;
// ADC conversion clock
ADCON2bits.ADCS = 0b110; // Fosc/64
// voltage reference
ADCON1bits.PVCFG = 0; // AVDD
ADCON1bits.NVCFG = 0; // AVSS
// input channel
ADCON0bits.CHS = 4; // CH4 (RA5)
// result format
ADCON2bits.ADFM = 1; // Left Justified
// acquisition delay
ADCON2bits.ACQT = 0; // Manual acquisition
// turn on ADC
ADCON0bits.ADON = 1; // Turn on ADC
// wait for acquisition
__delay_ms(1); // Delay 1ms
```

Then use the following code to retrieve the ADC output and store it in the variable ADCvalue:

```
// Start ADC conversion
ADCON0bits.GO = 1;
// wait for the conversion to finish
while(ADCON0bits.GO);
// read ADC
ADCvalue = (ADRESH << 8 | ADRESL);
```

The ADC channel is available on the datasheet (section 18.2) for the PIC18F45K50:

REGISTER 18-1: ADCON0: A/D CONTROL REGISTER 0

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	CHS<4:0>					GO/$\overline{\text{DONE}}$	ADON
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **Unimplemented:** Read as '0'

bit 6-2 **CHS<4:0>: Analog Channel Select bits**

00000 = AN0
00001 = AN1
00010 = AN2
00011 = AN3
00100 = AN4
00101 = AN5(1)
00110 = AN6(1)

Current sensor:
ADCON0bits.CHS = 0;
Potentiometer:
ADCON0bits.CHS = 1;
Light sensor:
ADCON0bits.CHS = 2;
Temperature sensor:
ADCON0bits.CHS = 3;
Thermocouple sensor:
ADCON0bits.CHS = 4;

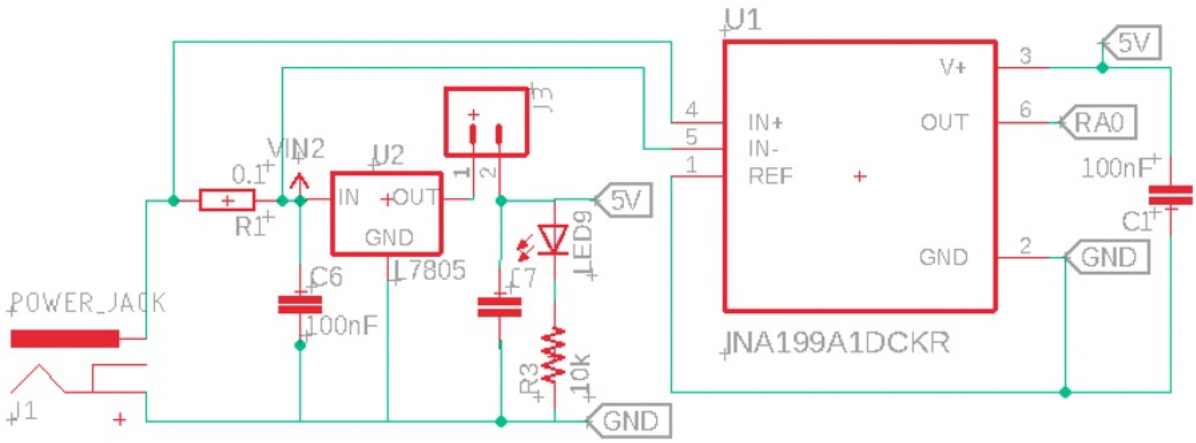
Power supply

The purpose of the built in power supply is to power the sensors and allow the user to monitor the power consumption of their project. More information on this can be found in the current sensor section of this document.

The analogue sensor board can be powered via the built in power supply or the GPIO headers. If you want to use the GPIO headers, the main PIC board must be powered via the DC input to provide power to this board. To use the current sensor, this board and the PIC board must be powered by the DC input on the analogue sensor board.

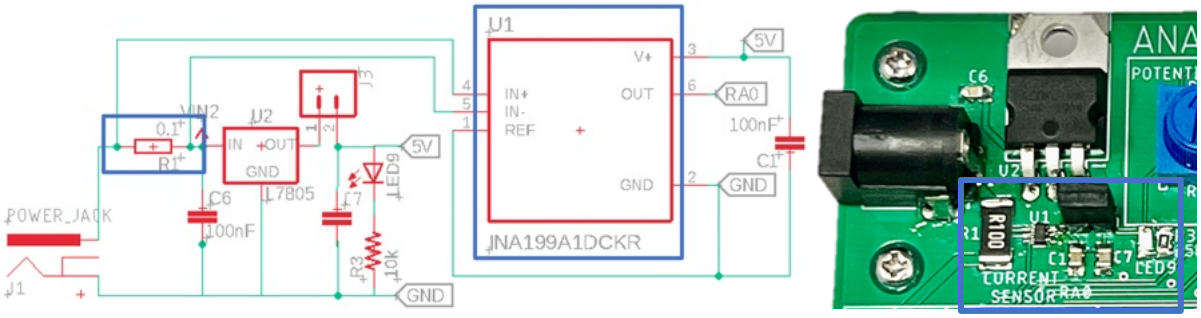
The Built in power supply requires a 2.1mm centre positive DC jack power supply that can provide 7-18V DC power with a minimum of 0.1 Amps, more power may be required for the add-on boards and accessories.

LED9 will illuminate if the 5V rail is working correctly.



Current Sensor

The current sensor utilises a 0.1Ohm 2W resistor and a INA199 Amplifier to enable the user to monitor the current coming through the analogue sensor board's power supply. This will output an analogue voltage on RA0 (ADCON0bits.CHS = 0) relative to the current.

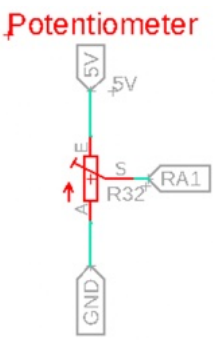


The raw ADC value will output the current in milliamps. To convert this to Amps, multiply the ADC value by 0.001.

```
// Start ADC conversion
ADCON0bits.GO = 1;
// wait for conversion
while(ADCON0bits.GO);
// read ADC
ADCvalue = (ADRESH << 8 | ADRESL) * 0.001;
```

Potentiometer

The blue dial generates a simple analogue voltage that can be easily varied by the user between 0V and 5V. The output voltage of the potentiometer will output on RA1 (ADCON0bits.CHS = 1).



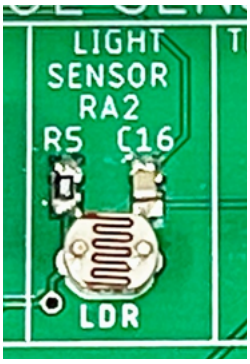
When read by the ADC on the PIC, the value will range from 0-1023. This can be converted into 0-100% by multiplying the ADC result by 0.098.

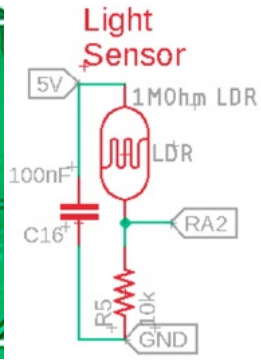
$100 \div 1023 = 0.098$

```
// Start ADC conversion
ADCON0bits.GO = 1;
// wait for conversion
while(ADCON0bits.GO);
// read ADC
ADCvalue = (ADRESH << 8 | ADRESL) * 0.098;
```

Light Sensor

The light sensor can measure the ambient light and can be varied by either covering or applying extra light to it. The output voltage of the light sensor will output on RA2 (ADCON0bits.CHS = 2).





When read by the ADC on the PIC, the value will range from 0-1023. This can be converted into 0-100% by multiplying the ADC result by 0.098.

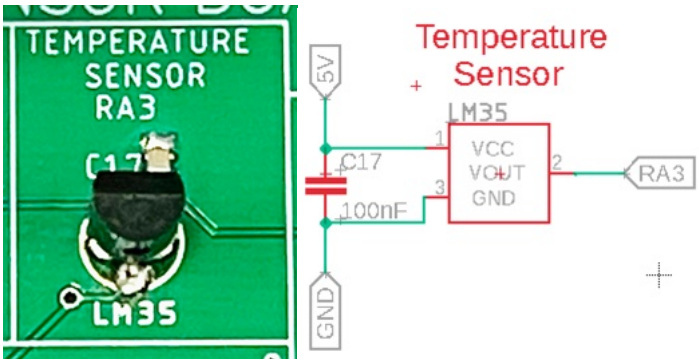
$100 \div 1023 = 0.098$

$512 \times 0.098 = 50.176 \approx 50\%$

```
// Start ADC conversion
ADCON0bits.GO = 1;
// wait for conversion
while(ADCON0bits.GO);
// read ADC
ADCvalue = (ADRESH << 8 | ADRESL) * 0.098;
```

Temperature Sensor

The temperature sensor measures the ambient temperature. The output voltage of the temperature sensor will output on RA3 (ADCON0bits.CHS = 3).



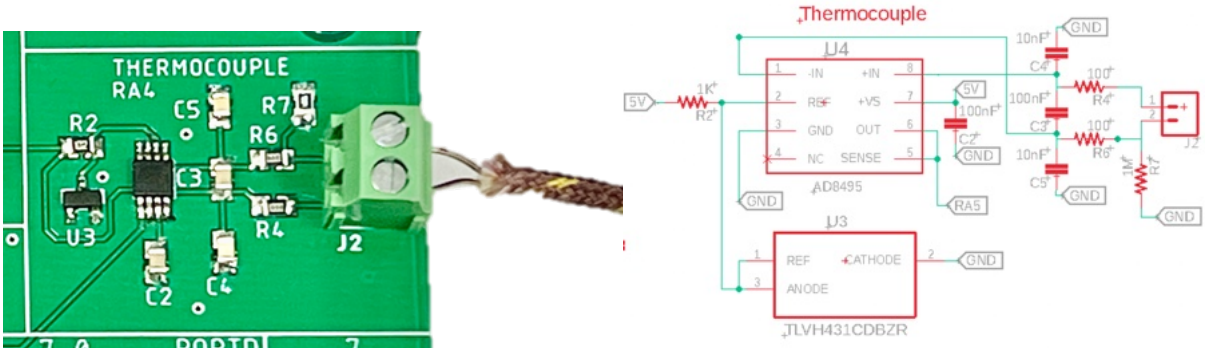
When read by the ADC on the PIC, the value will range from 0-1023. This can be converted into Centigrade by multiplying the ADC result by 0.489

Convert the voltage range into millivolts
$5v * 1000 = 5000mV$
Divide the millivolt range by the 10-bit ADC to get the millivolt per bit
$5000 \div 1023 = 4.8875855327$
The LM35 temperature sensor converts temperature to voltage like this:
$10mV\ Per\ °C$
$10mV \times 20°C = 200mV$
$200 \div 4.89 = 40.92$
$40.92 \times 0.489 = 20°C$

```
// Start ADC conversion
ADCON0bits.GO = 1;
// wait for conversion
while(ADCON0bits.GO);
// read ADC
ADCvalue = (ADRESH << 8 | ADRESL) * 0.489;
```

Thermocouple Sensor

The thermocouple sensor uses a K-Type thermocouple connected into J2. This enables the user to precisely measure temperature at the weld on the end of the thermocouple. The output voltage of the thermocouple sensor will output on RA5 (ADCON0bits.CHS = 4).



To read the temperature on the ADC we need to do a conversion.

Divide the voltage range by the 10-bit ADC to get the volts per bit
$5 \div 1023 = 0.0048875855327$
The ADC output will need to be multiplied by this to convert it into volts

The thermocouple sensor converts temperature to voltage like this:

$$\frac{(Vout - 1.25)}{0.005} = Temperature\ in\ Celsius$$

Here is the complete equation:

$$\frac{(ADC * 0.00488) - 1.25}{0.005} = Temperature\ in\ Celcius$$

For example:

$$\frac{(277 * 0.00488) - 1.25}{0.005} = 20.352°C$$

```
// Start ADC conversion
ADCON0bits.GO = 1;
// wait for conversion
while(ADCON0bits.GO);
// read ADC
ADCvalue = (((ADRESH << 8 | ADRESL)*0.00488) - 1.25) / 0.005;
```