DESIGNING A PID CONTROLLER USING ANALOG INPUT AND ANALOG OUTPUT SIGNALS USING DIGITAL AND ANALOG COMPONENTS FOR LABORATORY USE

NOVEMBER 16, 2019

PRANABENDRA PRASAD CHANDRA
BEE – IV, Section B2
Roll No. 001610801122

Acknowledgement

The success and final outcome of this project required a lot of guidance and assistance from many people and I am extremely privileged to have got this all along the completion of my project. All that I have done, is only due to such supervision and assistance and I would not forget to thank them.

I wish to express my heartfelt gratitude to my parents, for their encouragement, moral support and care.

I am obliged and grateful to my professors Prof. T.K. Ghoshal, Prof. S. Bhattacharya and Prof. S. Sadhu for providing me the opportunity to complete a project work on the topic '**Designing a PID controller using analog input and analog output signals using digital and analog components for laboratory use'**. I also wish to express my gratitude to them for their constant support and guidance throughout the work.

This project helped me to understand digital PID controller and its thorough functioning. It also provided me an opportunity for self-learning and know how both hardware and software play equally important roles in the development of the product.

Also, I would like to extend my sincere thanks to all my classmates for their constant help throughout the work.

Pranabendra Prasad Chandra

Contents

ABSTRACT

Project Topic :

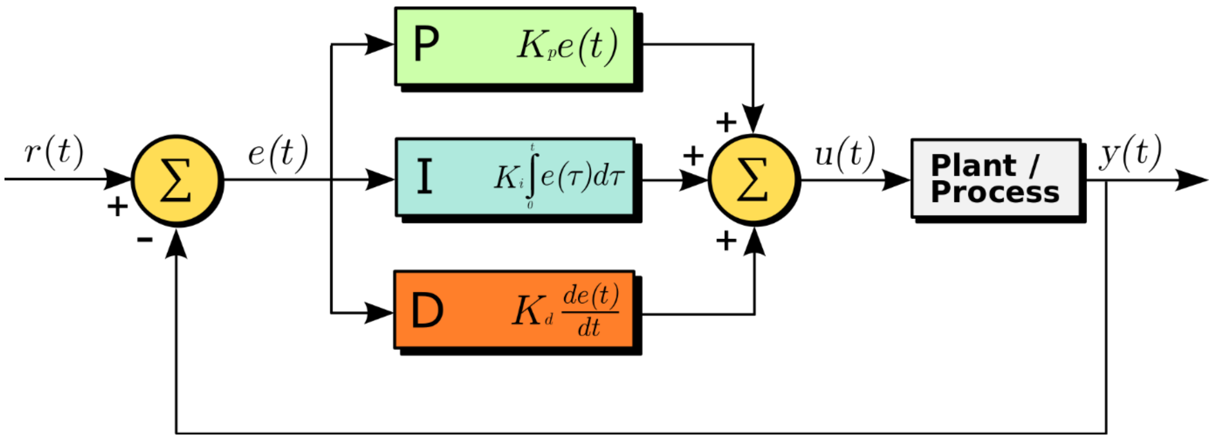
Designing a PID controller using analog input and analog output signals using digital and analog components for laboratory use

PID controllers can be used to regulate flow, temperature, pressure, level, and many other industrial process variables.

This project deals with the development of a digital PID controller for laboratory use. The controller takes in analog signal input, does the computations digitally and then outputs analog signal for the plant. The user operating this controller, can see the PID gains and adjust them as desired. Protections like overvoltage protection and latching of inputs have been incorporated, as a safety measure.

PID Controller

A PID controller calculates an error value **e** as the difference between a desired setpoint **r** and a measured process variable **y**. It then applies a correction based on the proportional, integral, and derivative terms. Hence, it enables automatic control.



^ Image source: *"PID controller overview" by Arturo Urquizo, PID Controller - Wikipedia*

PID in continuous-time domain

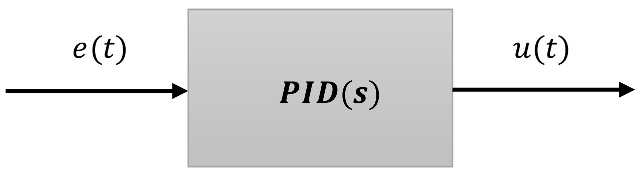
In continuous time domain, the PID equation is given as,

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t)$$

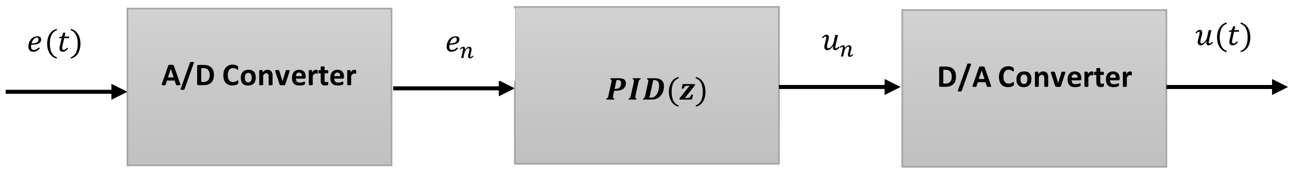
where,

- $e(t)$ is controller input
- $u(t)$ is controller output
- K_p is proportional gain
- K_i is integral gain
- K_d is derivative gain

In Laplace domain, the PID operation can be represented as,



PID in discrete-time domain



Let, the sampling frequency be f_s

$\therefore$ Sampling time period = $T_s = 1/f_s$

Discretizing the continuous time domain PID equation,

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt}e(t)$$

$$u_n = K_p e_n + K_i T_s \sum_{j=0}^{n-1} e_j + \frac{K_d}{T_s}(e_n - e_{n-1})$$

Since we have tried to simulate the digital PID controller in Simulink first, we have followed the transfer function offered by Simulink, all throughout the project.

The Discrete PID Controller block in Simulink, MATLAB has the transfer function,

$$\frac{U(z)}{E(z)} = K_p + K_i T_s \frac{1}{z-1} + K_d \frac{N}{1 + NT_s \frac{1}{z-1}}$$

Converting this transfer into difference equations, we obtain,

- $u_n^{(1)} = K_p e_n$
- $u_n^{(2)} = u_{n-1}^{(2)} + K_i T_s e_{n-1}$
- $u_n^{(3)} = u_{n-1}^{(3)}(1 - NT_s) + DN(e_n - e_{n-1})$

Summing these three difference equations, the controller output becomes

$$u_n = u_n^{(1)} + u_n^{(2)} + u_n^{(3)}$$

The filter coefficient N in the above transfer function implements derivative action, by forming a proper transfer function. If N is sufficiently large, the derivative term will tend to

$$\frac{K_d}{T_s}(z-1)$$

Implementation

Controller Specifications

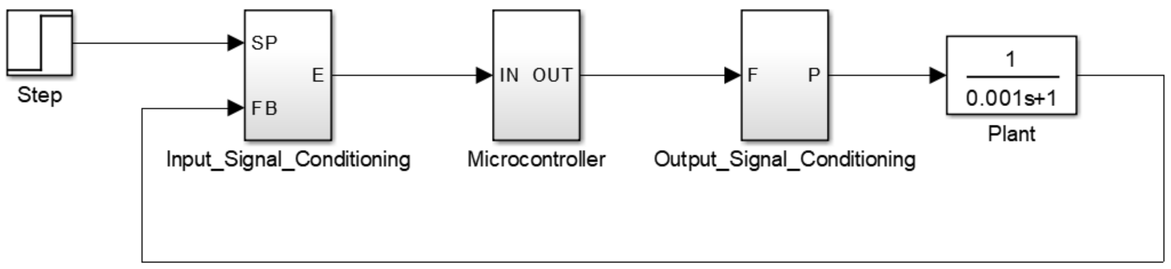
Supply voltage +12V and -12V (generated using two SMPS)

Voltage of input signal and feedback should be within +10V to -10V

Sampling frequency = 5000 Hz

PWM switching frequency = 100 kHz

Block Diagram and Simulation in Simulink



SP	:	Set Point
FB	:	Feedback (Plant Output)
E	:	Modified Error
IN	:	Microcontroller Signal Input
OUT	:	Microcontroller Signal Output (PWM)
F	:	Filter Input
P	:	Plant Input (Controller Output)

Input Signal Conditioning

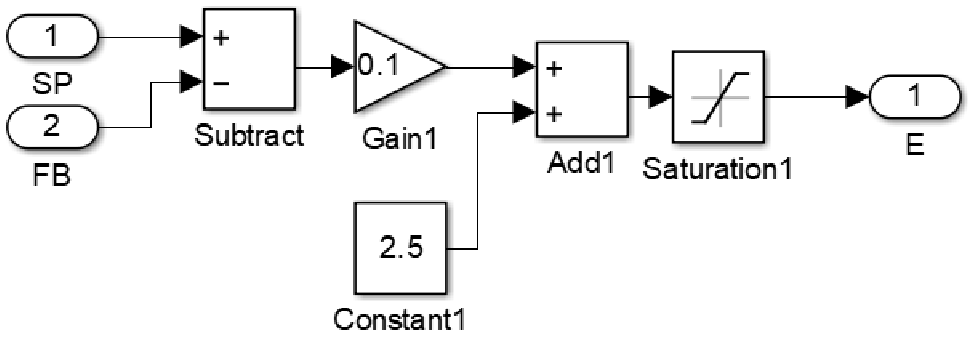
As mentioned in the controller specifications, $-10V \leq V_{SP}, V_{FB} \leq 10V$

Hence, $-20V \leq (V_{SP} - V_{FB}) \leq 20V$

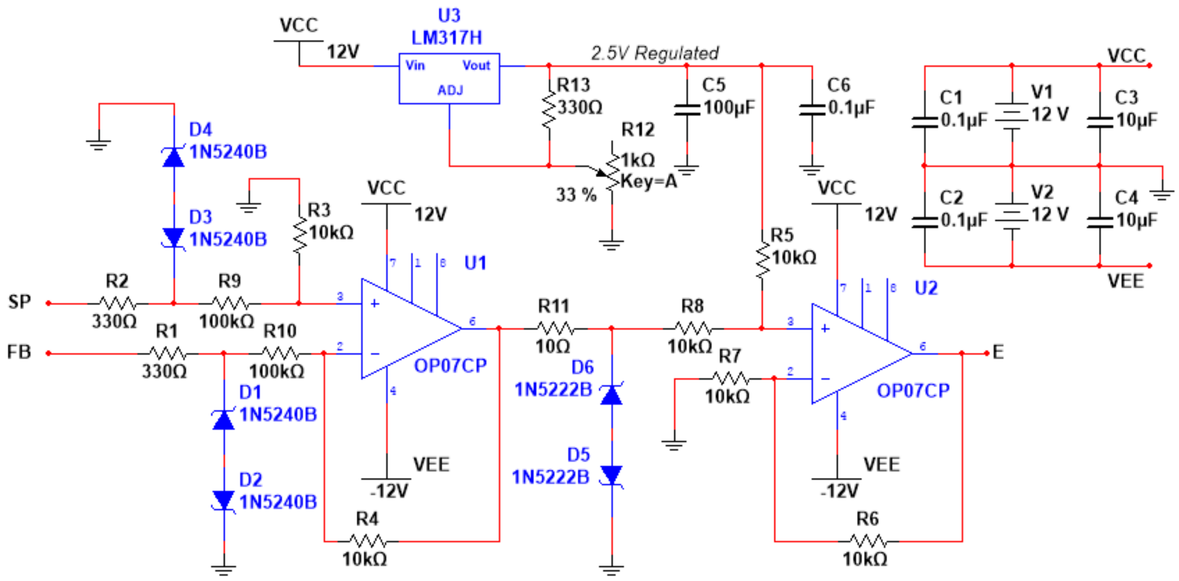
Modified error $E = \frac{V_{SP} - V_{FB}}{10} + 2.5V$

Therefore, $0.5V \leq E \leq 4.5V$

This bound of E ensures that the ADC of the microcontroller does not get damaged, as it can only take positive voltages less than or equal to 5V.

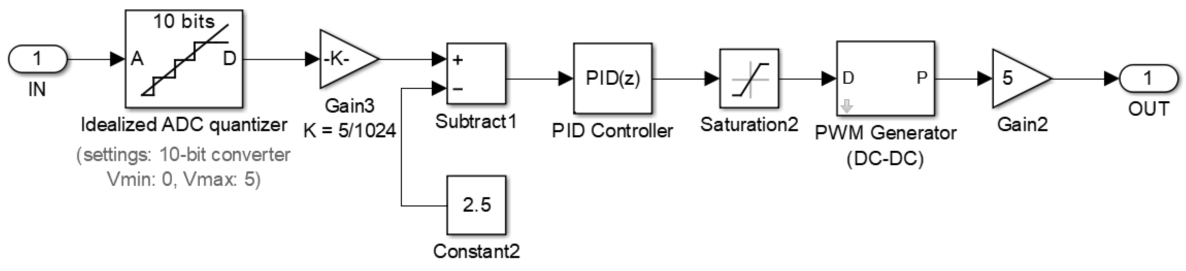


Circuit Realisation

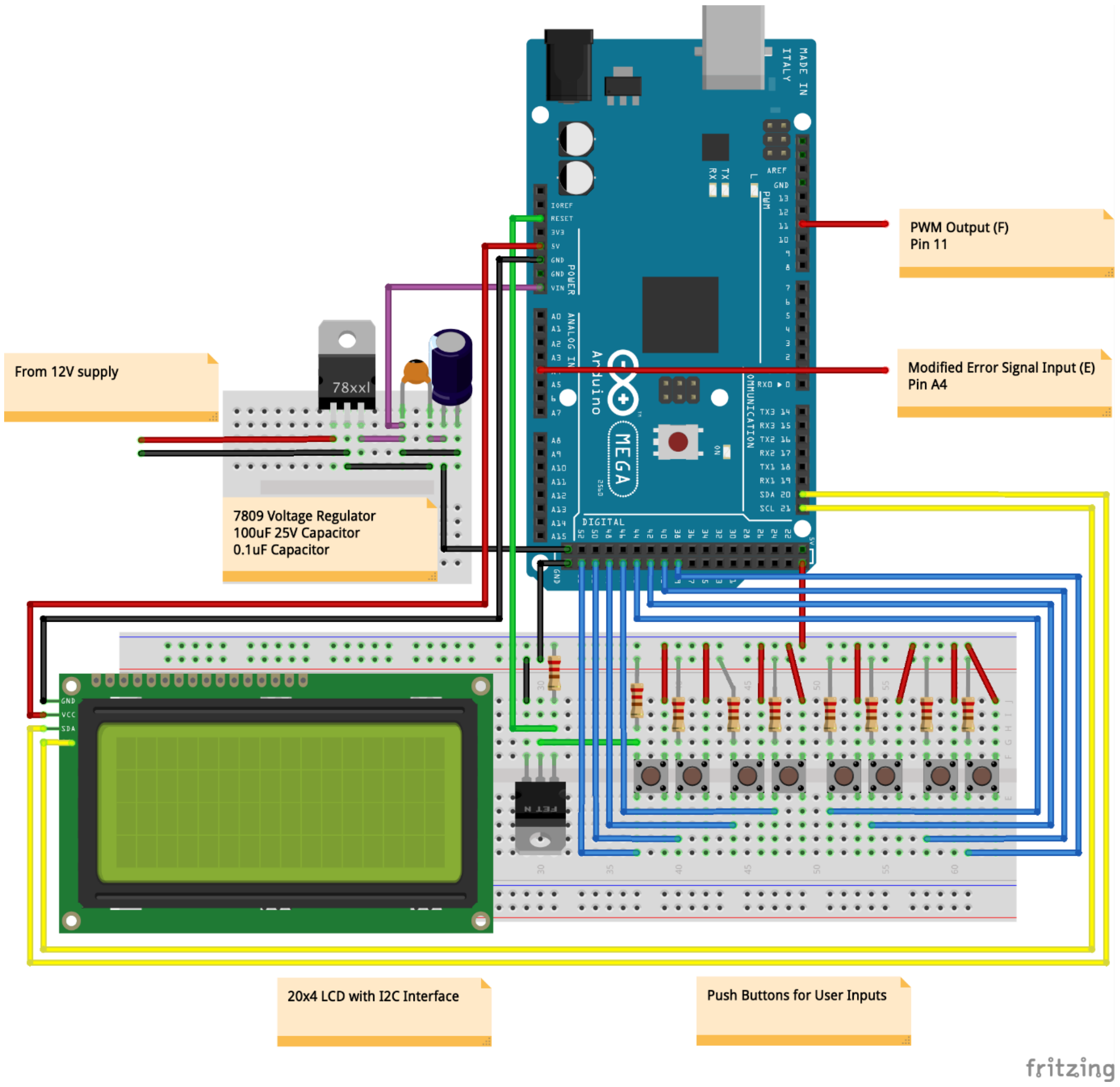


- U1 and U2 are precision op-amps, having very low offset voltage, in the order of microvolts.
- U1 acts as a differential amplifier having gain 0.1 to attenuate ($V_{SP} – V_{FB}$) by 10 times. D1 and D2 are used in back-to-back mode for overvoltage protection. Their breakdown voltage is 10V. This ensures that the signal passing through to R10 remains within +10V and -10V. Else, the Zener diodes clips the voltage. R1 is used as a current limiting resistance, to protect D1 and D2 from overcurrent. Similar is the case for D3, D4, R2 and R9.
- D5 and D6 have breakdown voltage of 2.5V. They function similarly, to keep the signal within the limits of -2.5V to +2.5V.
- U3 is an adjustable positive voltage regulator. R12 is adjusted finely, to set the output regulated voltage to 2.5V.
- U2 acts as a non-inverting adder, with gain 1. It generates an offset of 2.5V to the bipolar signal, to make it a unipolar signal (lying between 0V to 5V).
- All the capacitors used in the above circuit, are decoupling capacitors used with voltage supplies.

Microcontroller Operation

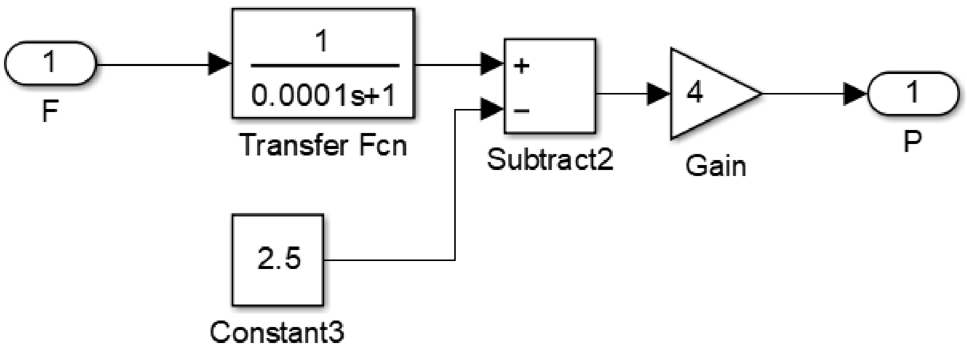


Circuit Realisation

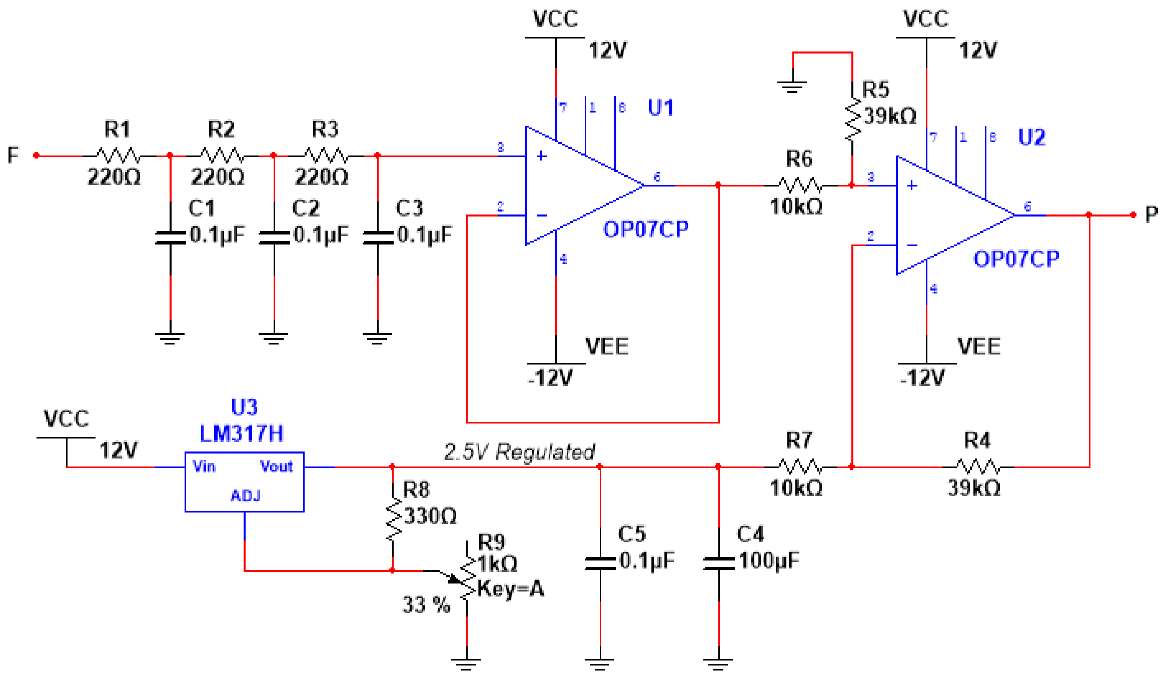


The microcontroller used in the project is Arduino MEGA 2560. It is a 5V tolerant, having 10-bit ADC and PWM output. The 20x4 LCD is used with I2C interface, to decrease the number of I/O pins. The push buttons are for user inputs. The LCD is for user to see and select the different PID parameters. The Arduino reads analog signal (modified error signal) and computes the output, which is modulated by the PWM output.

Output Signal Conditioning



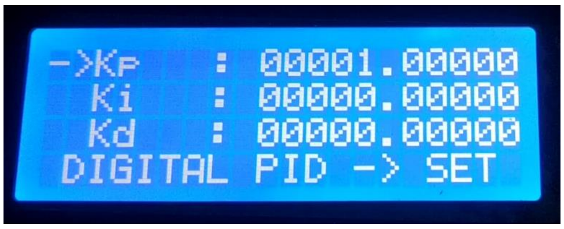
Circuit Realisation



- The PWM output from the microcontroller is passed through a three-stage cascaded RC low pass filter (cut-off frequency of approximately 1.5kHz), to obtain the demodulated controller output.
- U1 acts as a buffer to avoid loading effect.
- U2 as a differential amplifier, with gain 3.9 (approx. 4). The demodulated controller output is cleared of any offset, by subtracting 2.5V from it. Therefore, it converts the unipolar signal, to a bipolar signal.
- U3 is an adjustable positive voltage regulator. R12 is adjusted finely, to set the output regulated voltage to 2.5V.
- Capacitors C4 and C5 are used as decoupling capacitors along with the voltage supply.

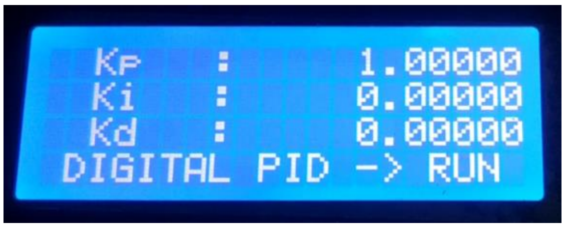
User Interface

LCD



```
->Kp   : 00001.00000
  Ki   : 00000.00000
  Kd   : 00000.00000
DIGITAL PID -> SET
```

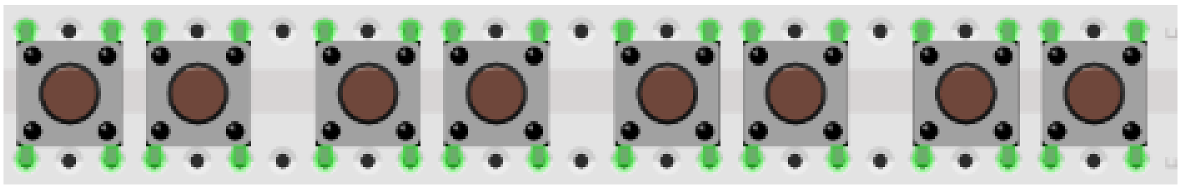
SET Mode



```
  Kp   :      1.00000
  Ki   :      0.00000
  Kd   :      0.00000
DIGITAL PID -> RUN
```

RUN Mode

Push Buttons (User Input)



RESET CLEAR SET RUN SELECT UP DOWN LEFT RIGHT

Button Name	Function	SET Mode	RUN Mode
RIGHT	Moves the cursor right	✓	
LEFT	Moves the cursor left	✓	
UP	Increments the current digit MOD 10	✓	
DOWN	Decrements the current digit MOD 10	✓	
SELECT	Selects parameter (K_p, K_i, K_d) to be set	✓	
SET/RUN	To change parameter value and update to run	✓	✓
CLEAR	Clears the parameter value to all zeros	✓	
RESET	Resets the microcontroller	✓	✓

Software Routines

Reading the states of the push buttons
Whenever the program enters **void loop()** the states of the eight push buttons are read and stored. If any button is pressed, the microcontroller will read HIGH, else it will read LOW.

Latching of button press
Whenever a button is pressed, it will be latched, but no HIGH state will be registered. During that time, if any other button is pressed, they will not be active to register their states. Only when the initial button is released, the falling edge is detected and HIGH is registered for that particular button only.

Detecting which button is pressed
From the previous state and the present state of push buttons, the button that is pressed, is detected. For that particular button, necessary action as a result of the button press.

Display LCD in SET mode
When the controller is in SET mode, the parameter pointer blinks and the cursor blinks for the user to change or update the value.

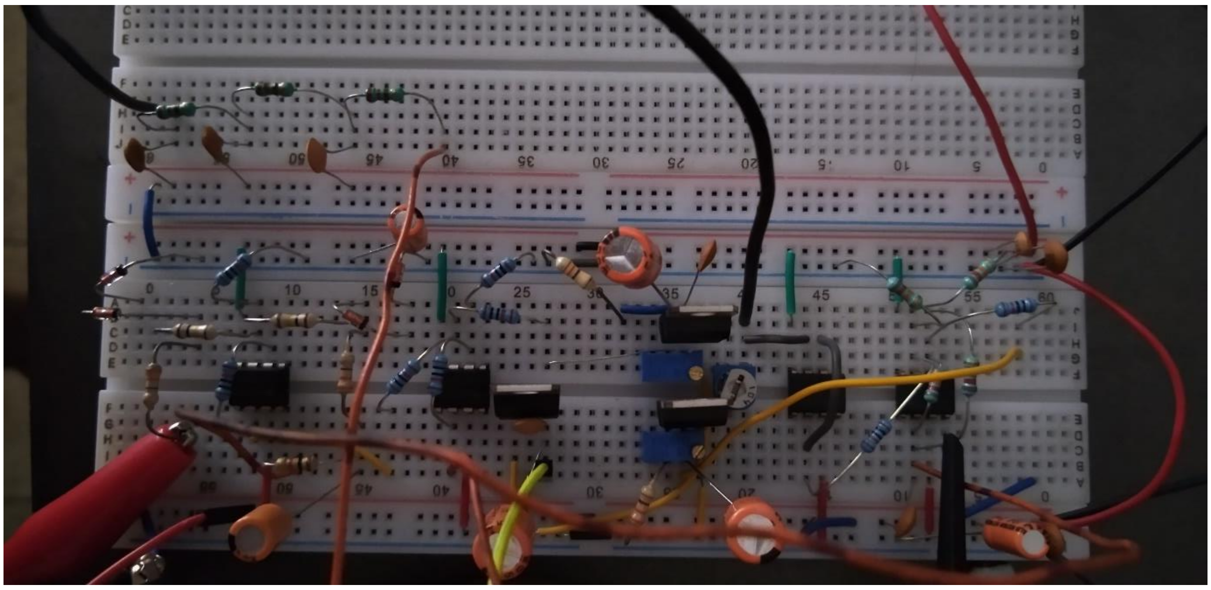
Display LCD in RUN mode
When the controller is in RUN mode, the LCD becomes static and shows the current value of the parameters.

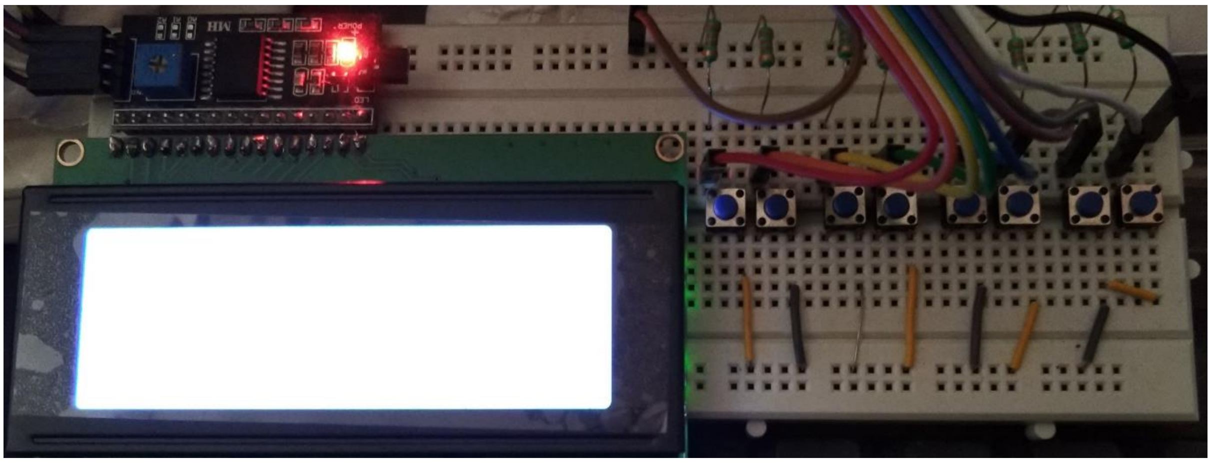
Reading the error value
The modified error signal is read and the offset is removed to get the attenuated error value.

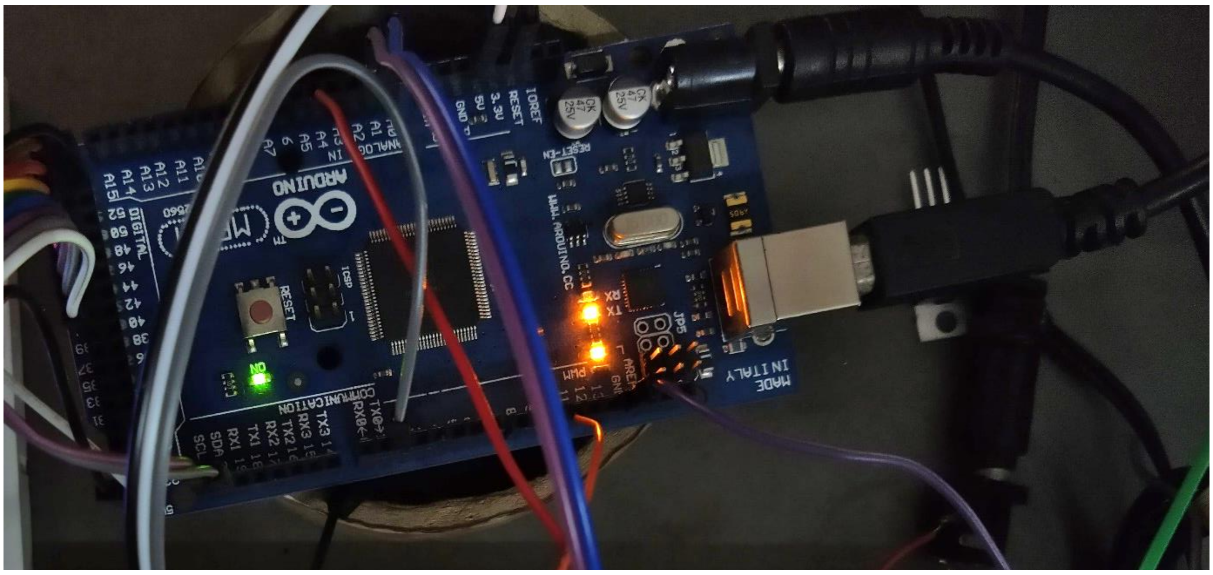
Computing the controller output
The previous controller output, the present sample and the previous sample of the error is passed through a set of difference equations to obtain the current controller output.

Changing the duty cycle of PWM
The current controller output value determines the duty cycle of the PWM, which is always running at constant switching frequency.

Experimental Setup





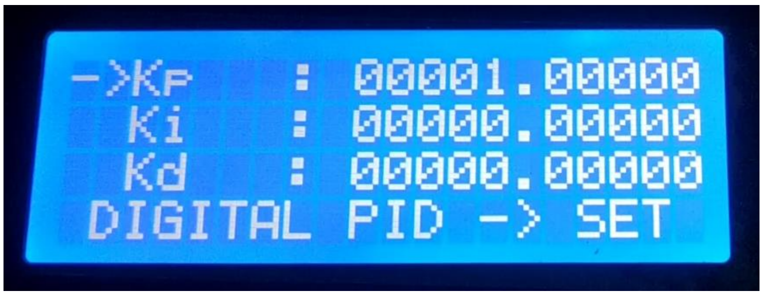


Top to bottom: Signal conditioning circuits, Push buttons and LCD, Arduino MEGA

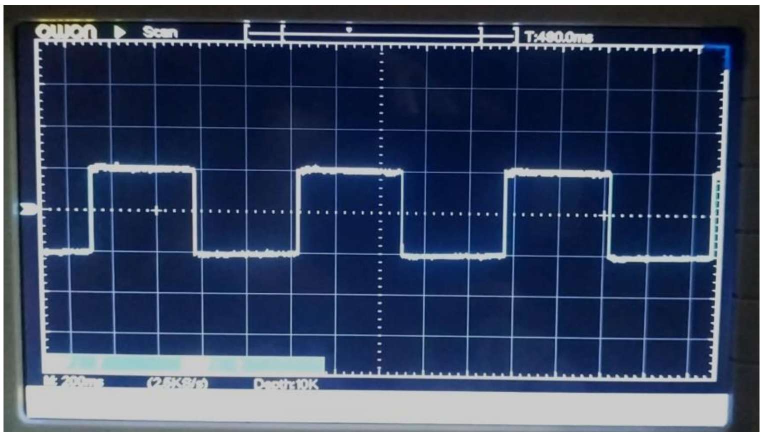
Result

Open Loop

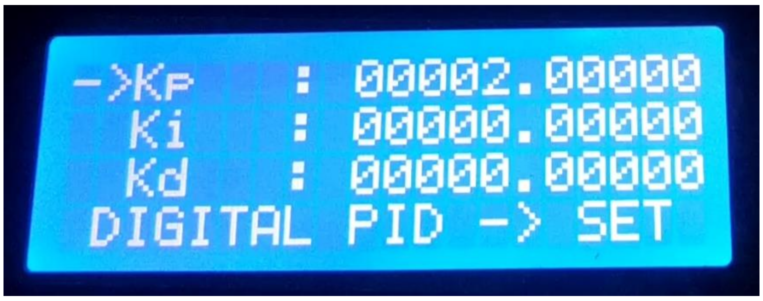
Kp = 1



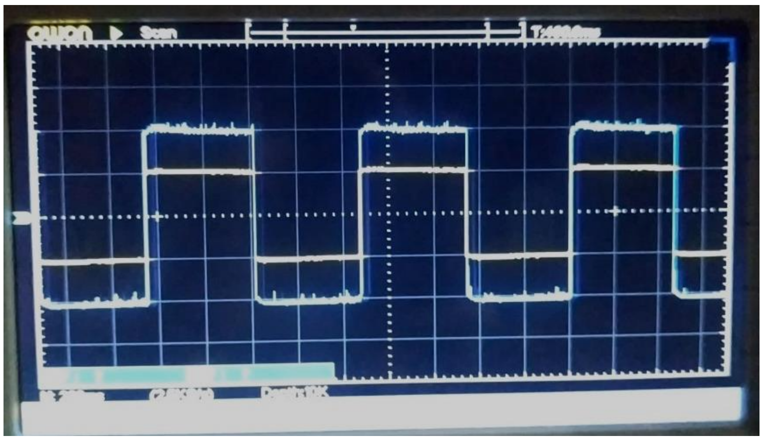
```
->Kp    : 00001.00000
  Ki    : 00000.00000
  Kd    : 00000.00000
DIGITAL PID -> SET
```



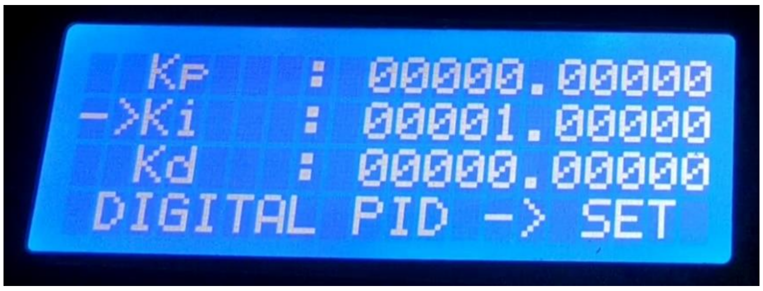
Kp = 2



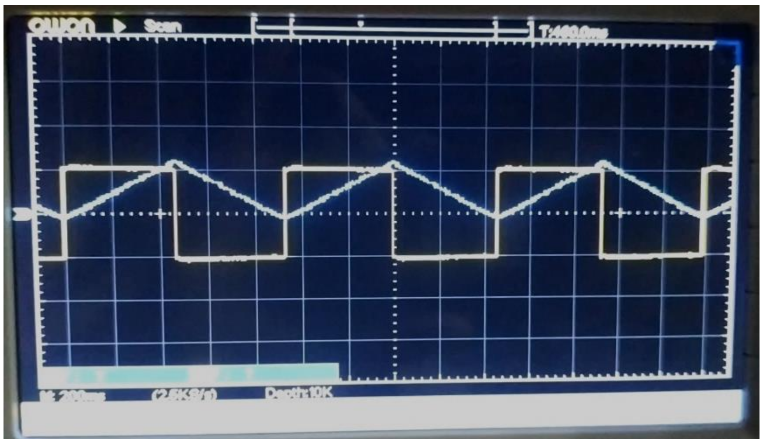
```
->Kp    : 00002.00000
  Ki    : 00000.00000
  Kd    : 00000.00000
DIGITAL PID -> SET
```



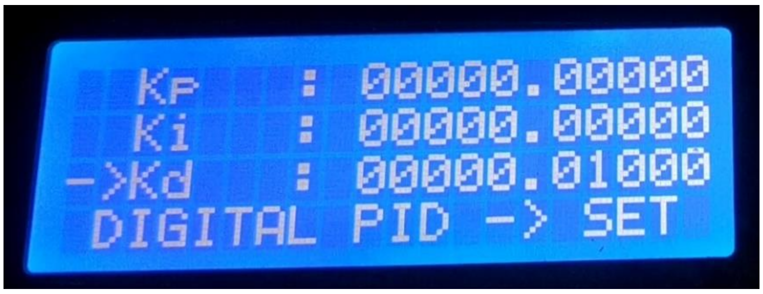
Ki = 1



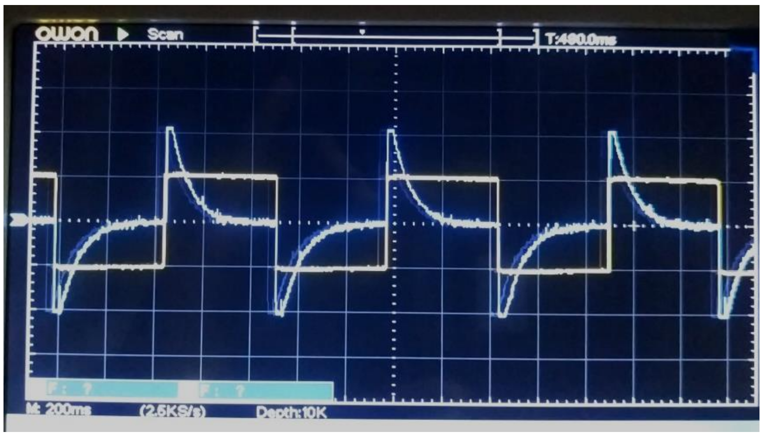
```
  Kp    : 00000.00000
->Ki    : 00001.00000
  Kd    : 00000.00000
DIGITAL PID -> SET
```



Kd = 0.01

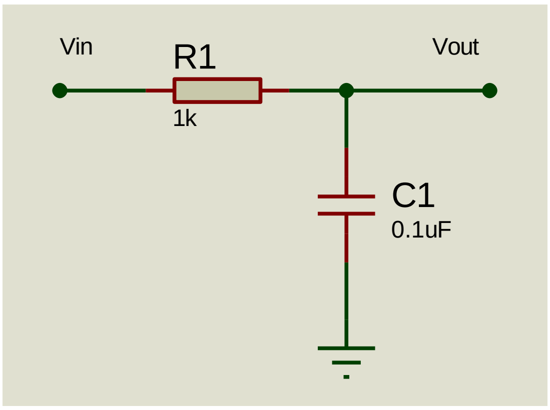


```
  Kp    : 00000.00000
  Ki    : 00000.00000
->Kd    : 00000.01000
DIGITAL PID -> SET
```



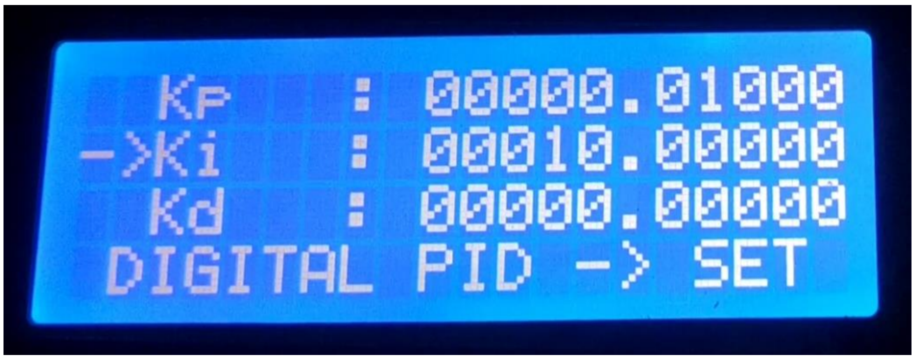
Closed Loop

Plant

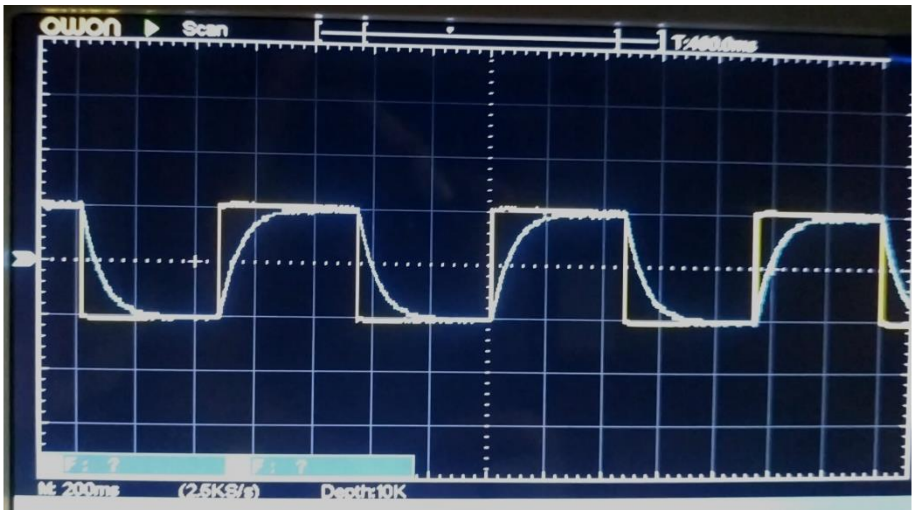


$$Plant\ Transfer\ Function : \frac{1}{0.0001s + 1}$$

Kp = 0.01, Ki = 10 (PI control)

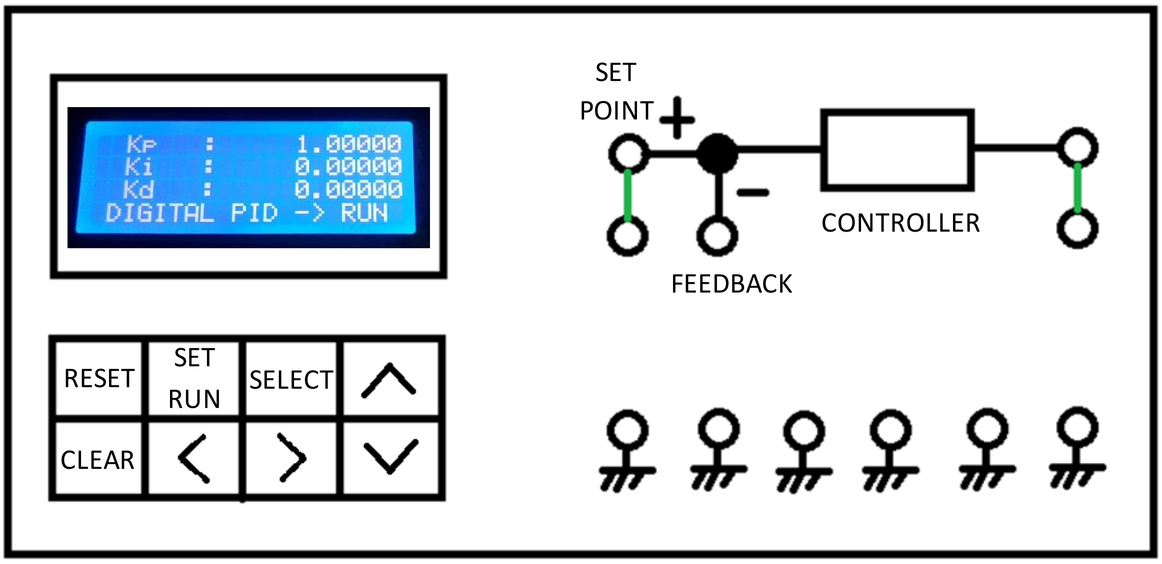


```
  Kp    : 00000.01000
->Ki    : 00010.00000
  Kd    : 00000.00000
 DIGITAL PID -> SET
```



Proposed Box Design

FRONT VIEW



Conclusion

1. Due to the use of passive RC filter to demodulate the controller output from PWM, the controller performs poorly if the set point signal frequency becomes comparable to the cut off frequency of the filter. Also, the system time constant increases slightly. These effects can be eliminated, if a microcontroller with an inbuilt DAC is used.

2. This digital PID controller works as desired for low frequencies (<= 100 Hz).

3. The controller is the amalgamation of both hardware and software, giving desired output as set by user via the user interface.

References

- Shinskey, F.G., 1979. *Process control systems*. McGraw-Hill, Inc..
- Johnson, C.D., 1999. *Process control instrumentation technology*. Prentice Hall PTR.
- Boylestad, R.L. and Nashelsky, L., 2012. *Electronic devices and circuit theory*. Prentice Hall.
- Carter, B. and Brown, T.R., 2001. *Handbook of operational amplifier applications*. Dallas, Tex, USA: Texas Instruments.
- Shirriff, K. and Badger, P., 2009. Secrets of Arduino PWM. *Ken Shirriff's Blog*.
- How to control a character I2C LCD with Arduino by Benne de Bakker https://www.makerguides.com/character-i2c-lcd-arduino-tutorial/
- Wikipedia contributors, "PID controller," *Wikipedia, The Free Encyclopedia,* https://en.wikipedia.org/w/index.php?title=PID_controller&oldid=925590136 (accessed November 15, 2019).
- Arduino Forum, “PWM Frequency Library”, https://forum.arduino.cc/index.php/topic,117425.0.html

Appendix

List of Components Used

Component Type	Component Name	Functions
Microcontroller	Arduino Mega 2560 (Rev 3)	Processing unit
SMPS	12V, 1A	Voltage Source
LCD Module	20x4	LCD Display
I2C Interface for LCD	I2C Module	To reduce LCD I/O
Precision Op Amp	OP07	Very low input offset voltage
Constant Voltage Regulator	7809	To power Arduino
Variable Voltage Regulator	LM317	To generate offset
Zener Diodes	1N5240B, 1N5222B	Overvoltage protection
N Channel MOSFET	BS170	Switch to reset Arduino
Push Buttons	1825910-7	To take user inputs
Resistors	10, 220, 330, 1k, 10k, 100k, 39k (all in Ω)	
Potentiometers	1k Trim pot	For accurate setting of voltage
Capacitors	0.1, 10, 100 (all in µF)	