Innovative
Ghar Nepal

नेपाल बन्छ, जब नेपालमै बन्छ

Confidential

Project Report
On
Paani Mantra

Submitted by:
R&D Team

Submitted to:
Management Team,
Mantra, Chanting of Technology

List of Figures

Contents

Chapter 1

Introduction

1.1 Background

Almost every houses in Kathmandu valley has at least one overhead tank and one ground tank. Maintaining water level in those tanks is very important. Sometimes, there may not be water in overhead tank in emergency time. Sometimes, water in overhead tank may overflow. Sometimes, motor may be left turned on even if there is no water in ground tank. Monitoring water level in each tank and switching motor on or off are required. However, monitoring and controlling manually is boring job and sometimes we might forget to switch on or off motor in time. That might causes water overflow, extra electric bill etc.

Paani Mantra is a device that overcomes the above mentioned problem. It monitors and control motor automatically without need of any person. Manual control function is also available.

1.2 Paani Mantra

Paani mantra is electronic device that monitors and control water level in water tank automatically. It uses magnetic float sensor to sense water level in overhead tank as well as underground tank. It displays current status of water level in both tank and also turn on or off motor accordingly. It turns motor on if water level is below 25% and there is water is ground tank. If there is no water in ground tank, motor is turned off. Motor is turned off if overhead tank is full. Manual motor on is available that works if overhead tank is not full and there is water in ground tank. Status of overhead tank, ground tank and motor is displayed on Paani Mantra with led indicator.

1.3 Outline of Report

This is the final report of project *Paani Mantra.* It contains our approach in designing the device as well as result both in simulation and hardware.

This report also briefly presents about the main components used in the device.

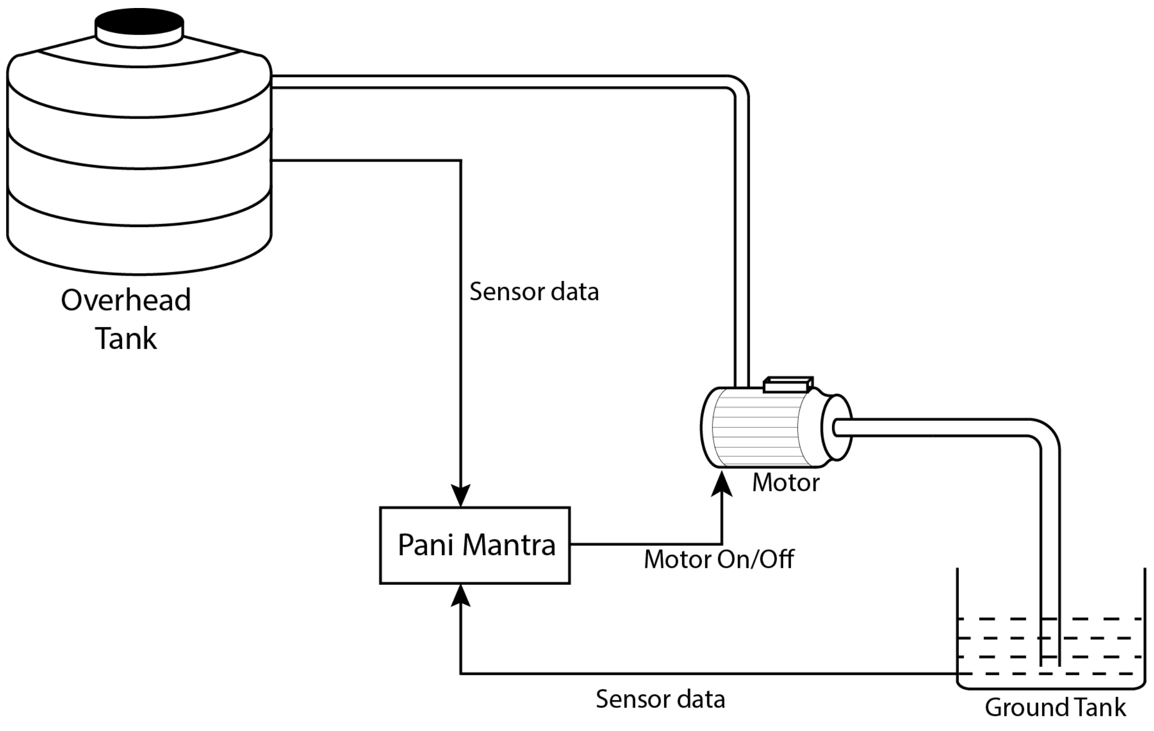


Figure 1.1: System layout of Paani Mantra.

Chapter 2

Algorithm Development and *MATLAB* Simulation

2.1 Software Tools

MATLAB was used for the simulation purposes and *Logic Friday* was used to generate logic equation and gate diagram.
Some features of *MATLAB* used in this project are:

1. Simulation of control logic in Simulink.
2. Embedded code generation from simulink model.

2.2 Algorithm Development

Following procedures were followed to develop algorithm:

1. Requirement of *Paani Mantra* was listed.
2. Truth tables for status of motor, led indicator were created using the requirement listed in previous step.
3. Using *Logic Friday*, minimized logic equation for every output was derived separately and combined gate diagram was generated.
4. The generated gate diagram with necessary modification (for sequential logic) was implemented in *Simulink*.
5. Some external model to make simulation interactive were added and simulated.

2.2.1 Requirement of Paani Mantra

Following are the requirements of *Paani Mantra:*

1. Motor should be on if overhead tank is empty and there is water in ground tank.
2. Motor should be off when overhead tank is full.
3. Motor should be off when there is no water in ground tank.
4. Motor can be turned on by emergency pushbutton if and only if overhead tank is not full and there is water in ground tank.

5. Motor is turned off after 10 second when overhead tank is full.
6. LEDs should turn on to show respective level of water.
7. If water in overhead tank is less than 25 %, bottom sensor status LED should blink.
8. If ground tank is empty, ground tank status LED should blink.

2.2.2 Truth Table

For the implementation of truth table, following convention were taken:

1. Input:
   - No water detected by sensor: LOW (0)
   - Water detected by sensor: HIGH (1)
   - Emergency push button not pressed: HIGH (1)
   - Emergency push button pressed: LOW(1)
2. Output:
   For output, HIGH signal means ON and LOW signal means OFF.

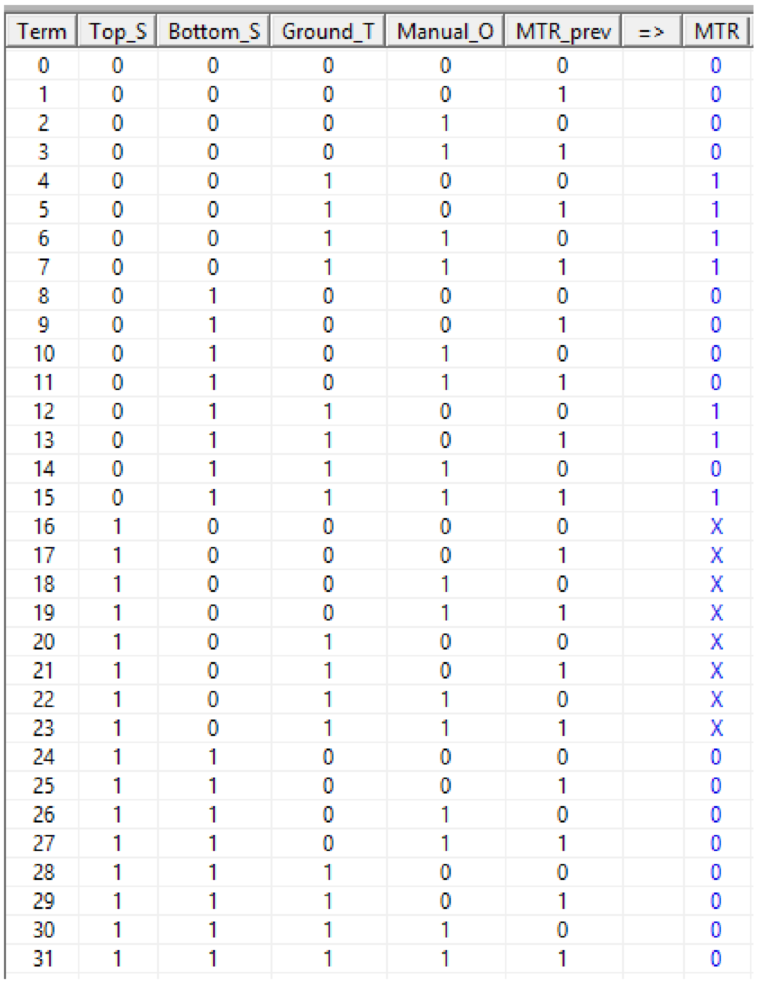
These convention were chosen based on economical design of hardware (Description of hardware is given in next chapter.) Convention of intermediate logic variable will be explained in respective section. Following variables has been defined for truth table:

- Top_S: Output of top sensor.(Overhead tank)
- $Middle\_S$: Output of middle sensor (Overhead tank).
- $Bottom\_S$: Output of bottom sensor (Overhead tank).
- $Ground\_T$: Output of ground tank sensor.
- $Manual\_O$: Emergency push button.
- MTR: Output control signal for motor.
- MTR_prev: Previous output control signal for motor (One sample before present state).
- Top_D, $Middle\_D$ and $Bottom\_D$: Status indicator for overhead tank.
- $Tank\_D$: Status indicator for ground tank.

Logic Friday was used to generate logic equation and gate diagram from truth table.

Truth table for motor control:

Here delay of 10 seconds (when overhead tank is full) is not implemented. It will be added at last. Output MTR depends on input $Top\_S, Bottom\_S, Ground\_T, Manual\_O$ and MTR_prev. Using *Logic Friday*, following logic equation was determined:



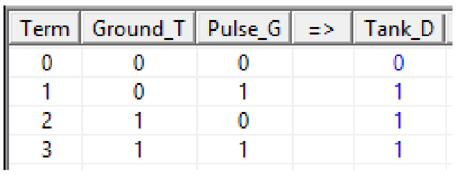
Term	Top_S	Bottom_S	Ground_T	Manual_O	MTR_prev	=>	MTR
0	0	0	0	0	0		0
1	0	0	0	0	1		0
2	0	0	0	1	0		0
3	0	0	0	1	1		0
4	0	0	1	0	0		1
5	0	0	1	0	1		1
6	0	0	1	1	0		1
7	0	0	1	1	1		1
8	0	1	0	0	0		0
9	0	1	0	0	1		0
10	0	1	0	1	0		0
11	0	1	0	1	1		0
12	0	1	1	0	0		1
13	0	1	1	0	1		1
14	0	1	1	1	0		0
15	0	1	1	1	1		1
16	1	0	0	0	0		X
17	1	0	0	0	1		X
18	1	0	0	1	0		X
19	1	0	0	1	1		X
20	1	0	1	0	0		X
21	1	0	1	0	1		X
22	1	0	1	1	0		X
23	1	0	1	1	1		X
24	1	1	0	0	0		0
25	1	1	0	0	1		0
26	1	1	0	1	0		0
27	1	1	0	1	1		0
28	1	1	1	0	0		0
29	1	1	1	0	1		0
30	1	1	1	1	0		0
31	1	1	1	1	1		0

Figure 2.1: Truth table for motor control signal (without 10 seconds delay).

$$MTR = Bottom\_S' \cdot Ground\_T + Top\_S' \cdot Ground\_T \cdot Manual\_O' + Top\_S' \cdot Ground\_T \cdot MTR\_prev$$

Truth table for Ground Tank status indicator:

$Pulse\_G$ is the intermediate variable whose value toggles every $500ms$. Value for this variable is generated internally in the microcontroller.



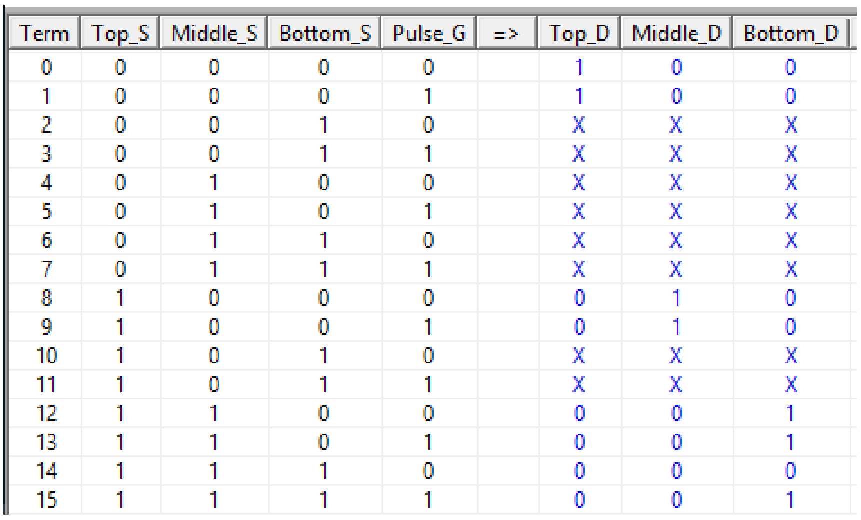
Term	Ground_T	Pulse_G	=>	Tank_D
0	0	0		0
1	0	1		1
2	1	0		1
3	1	1		1

Figure 2.2: Truth table for Ground Tank status indicator.

Logic equation was found to be:

$$Tank\_D = Ground\_T + Pulse\_G$$

Truth table for Overhead Tank status indicator:



Term	Top_S	Middle_S	Bottom_S	Pulse_G	=>	Top_D	Middle_D	Bottom_D
0	0	0	0	0		1	0	0
1	0	0	0	1		1	0	0
2	0	0	1	0		X	X	X
3	0	0	1	1		X	X	X
4	0	1	0	0		X	X	X
5	0	1	0	1		X	X	X
6	0	1	1	0		X	X	X
7	0	1	1	1		X	X	X
8	1	0	0	0		0	1	0
9	1	0	0	1		0	1	0
10	1	0	1	0		X	X	X
11	1	0	1	1		X	X	X
12	1	1	0	0		0	0	1
13	1	1	0	1		0	0	1
14	1	1	1	0		0	0	0
15	1	1	1	1		0	0	1

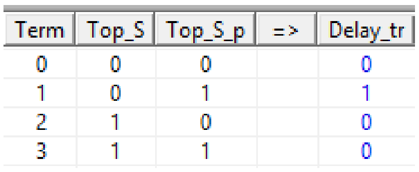
Figure 2.3: Truth table for Overhead Tank status indicator.

Logic equation was found to be:

$$Top\_D = Top\_S'$$
$$Middle\_D = Top\_S \cdot Middle\_S'$$
$$Bottom\_D = Middle\_S \cdot Bottom\_S' + Bottom\_S \cdot Pulse\_G$$

Truth table for Delay trigger signal:

This signal is generated to trigger 10 seconds delay when output of top sensor of overhead tank changes from LOW to HIGH. Here, Top_S_p represents previous state of top sensor of overhead tank.



Term	Top_S	Top_S_p	=>	Delay_tr
0	0	0		0
1	0	1		1
2	1	0		0
3	1	1		0

Figure 2.4: Truth table for delay trigger signal.

Logic equation was found to be:

$$Delay\_tr = Top\_S' \cdot Top\_S\_p$$

2.2.3 Gate Diagram

All the generated logic equation except that of delay trigger signal was combined and gate diagram was generated. Gate diagram for delay trigger signal was generated separately. They are combined in *MATLAB*.

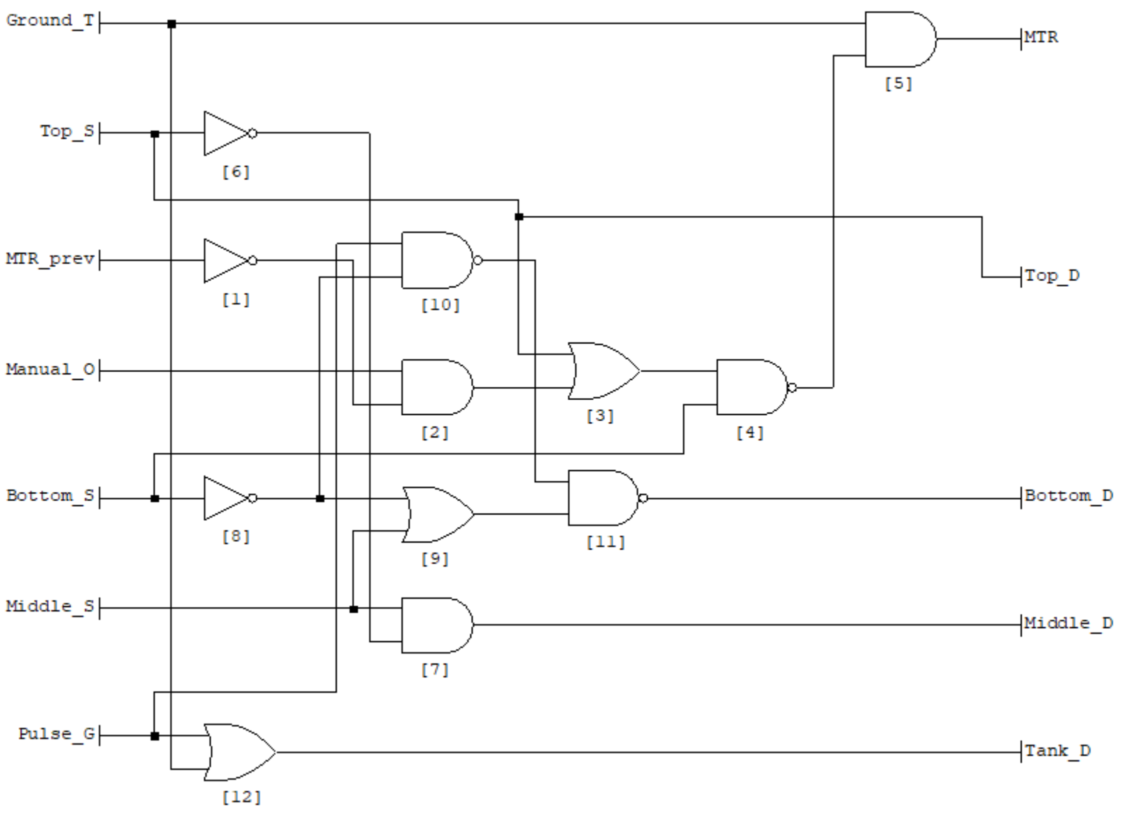


Figure 2.5: Gate diagram of main system without 10 seconds delay.

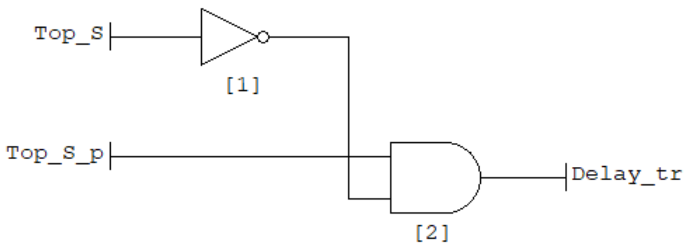


Figure 2.6: Gate diagram of delay trigger signal.

2.2.4 Implementing in *MATLAB*(Simulink)

Gate diagrams generated in previous step were added to *Simulink* and proper connection for signal was made. Positive edge triggered **Monostable** of 10 seconds was added to the output of delat trigger signal. Output of monostable is high for 10 seconds when delay is triggered. It can only be retriggered after 10 seconds.

The pseudocode for monostable is as follows:

```
if (delay_tr==TRUE and output == LOW)
{
tmp = 10;
}

if (tmp>0)
{
output = HIGH;
tmp = tmp - 0.005;      //Fundamental sample time was chosen to be 0.005 s.
}
else
{
output = LOW;
}
```

Output of delay signal *ORed* with *MTR* signal gives actual control signal for motor. Some external

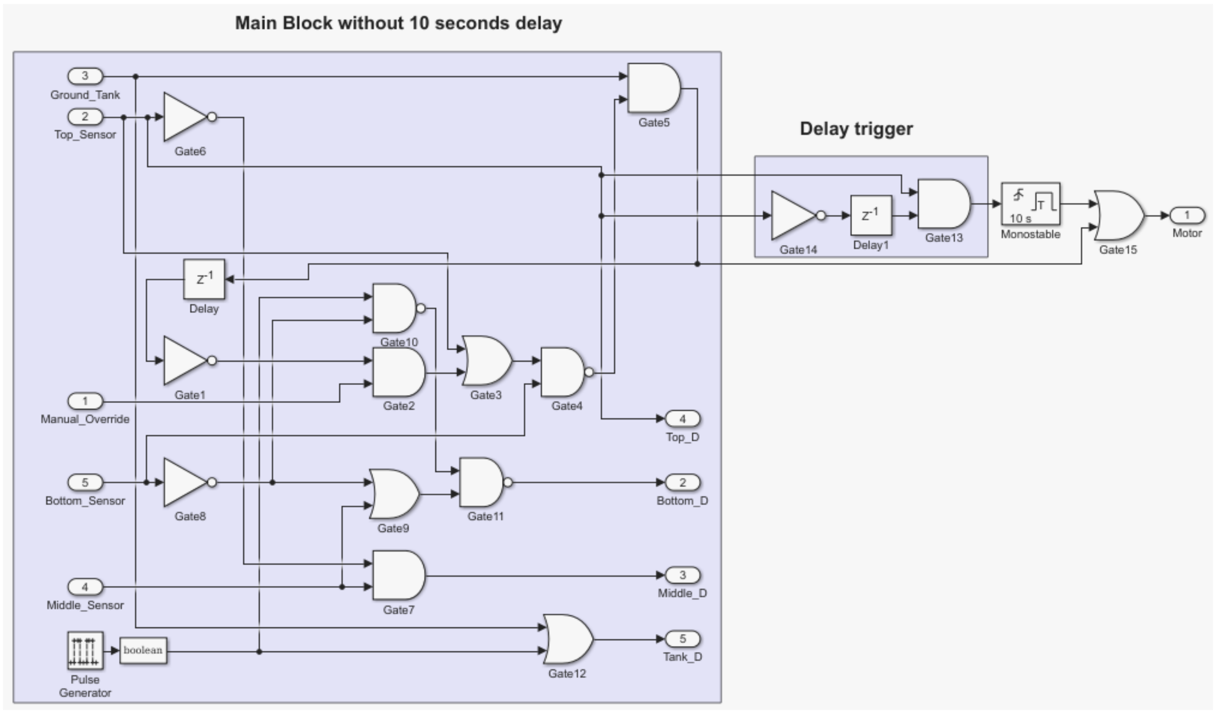


Figure 2.7: Simulink block of complete controller.

models were added to make simulation interactive. Switch is used to simulate the sensor and button is used to simulate emergency push button. Final simulink model is given in the next page.

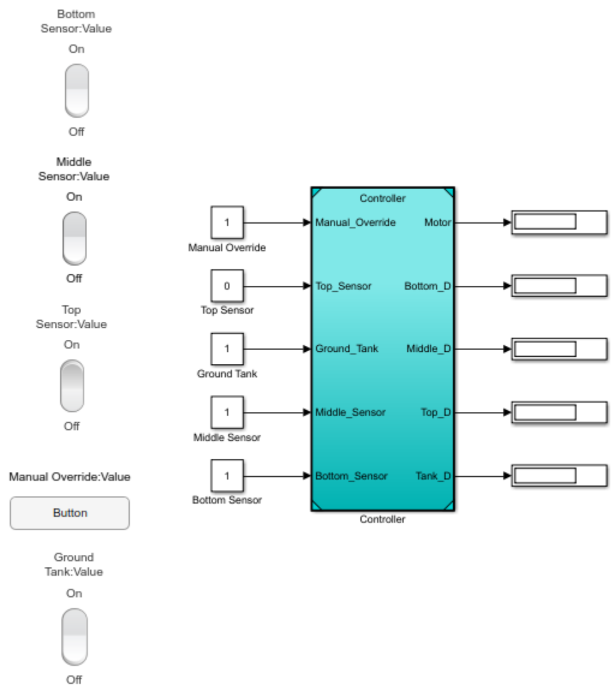


Figure 2.8: Simulink model of Paani Mantra.

Simulink model was simulated and results was found as per the requirement.

Chapter 3

Programming and Hardware

3.1 Software Tools

Proteus and *Autodesk EAGLE* was used for hardware design. *Atmel Studio* was used to write program code and *Extreme Burner AVR* is used to burn program to microcontroller. Simplified hardware circuit was created in *Proteus* and PCB layout was created in *Autodesk EAGLE*.
Some features of *Proteus* used in this project are:

1. Atmel AVR and other microcontroller can be simulated which can be used to test the program code.
2. Real-time analytical environment.

Some features of *Autodesk EAGLE* used in this project are:

1. Multisheet Schematics.
2. Powerful autorouting.
3. Electrical rule checking.
4. Design rule checking
5. Real-time synchronization between schematic and PCB layout.
6. User language programming.

3.2 Programming

3.2.1 Embedded Code Generation

Simulink supports code generation of the model. Embedded Coder was chosen for system target and Atmel's AVR (8 bit) was chosen for processor. Sample time of 0.005 seconds was defined. *C* was chosen as programming language. Finally, embedded *C* code was generated.

Generated code was added to Atmel Studio's project. Codes for initializing ports, taking input and giving output was written. Generated code was used for controller's arithmetic and logical operation. Code was compiled and hex file was generated.

3.2.2 Program Code

The *main* file of program is given below. Other files generated by *MATLAB* has not been presented here.

```
#include <avr/io.h>
#include "Extra/Controller.h"     /* Model's header file(within folder'Extra') */

int main(void)
{
int x;
/*Input/Output ports declaration.*/
DDRB = 0x00;                          /*Declare input port.*/
PORTB = 0xFF;                         /*Enable pull up resistor.*/
DDRD = 0xFF;                          /*Declare output port.*/

/*Setup and initialize timer1.*/
TCCR1B = (1<<WGM12)|(1<<CS10);        /*CTC mode, No pre-scalar.*/
TCNT1 = 0;
OCR1A = 39999;                        /*Value for 5ms.*/

/*Load initial state.*/
Controller_initialize();             /*Function generated by MATLAB.*/

while (1)
{
/*Input*/
x = PINB;
Controller_U.Bottom_Sensor = 0b00000001&x;
Controller_U.Middle_Sensor = (0b00000010&x)>>1;
Controller_U.Top_Sensor = (0b00000100&x)>>2;
Controller_U.Ground_Tank = (0b00001000&x)>>3;
Controller_U.Manual_Override = (0b00010000&x)>>4;

/*Logical Operation*/
Controller_step();                   /*Function generated by MATLAB.*/

/*Output*/
PORTD = (Controller_Y.Motor)|(Controller_Y.Motor<<3)|(Controller_Y.Bottom_D<<4)|
(Controller_Y.Middle_D<<5)|(Controller_Y.Top_D<<6)|(Controller_Y.Tank_D<<7);

while(!(TIFR&(1<<OCF1A)));          /*Loop till output compare flag is not set.*/
TIFR|=(1<<OCF1A);                   /*Reset output compare flag.*/
}
}
```

Model's header file has been included so that function generated by *MATLAB* can be used.

3.3 Hardware Design

3.3.1 Description of the component

1. **Magnetic Float Sensor**: This sensor is simply acts as on-off switch. Switch contact is held by magnet. When there is water, parts of sensor containing magnet floats and shift upward. So, switch contact is broken and acts as open. It can also be configured to make contact when there is water and break contact when there is no water simply by inverting the sensor. However, we have configured the sensor to make contact when there is no water and break contact when there is water.

2. **Push Button**: Push button was used for emergency motor on. Normally open type push button was used in this project.

3. **ATMega8**: ATMega8 is 8-bit AVR microcontroller manufactured by Atmel (now owned by Microchip). It uses advanced RISC architecture. Some features of ATMega8 used in this project are:

   (a) It has internal RC oscillator (Eliminating need of external oscillator).

   (b) It has 16-bit timer that can also be used in CTC (clear timer on compare match) mode.

   (c) It has internal pull-up resistors that can be enabled from software (Eliminating need of external pull-up resistors).

4. **BC547**: This is npn transistor that can be used for low power switching application. In this project, we've used this transistor to switch relay. Maximum collector current that can flow through this transistor is 100 mA. Maximum V_{CEO} is $45V$ with minumum of 110 current gain.

5. **Relay**: $12V$ relay has been used. Coil resistance was found to be 160Ω approximately. So the current drawn by relay is about $75mA$.

6. **LED**: 5 mm LEDs of different colour has been used. All LEDs forward drop voltage is nearly $2V$. Current rating of LEDs is $20mA$.

3.3.2 Calculation of component rating

1. **Base resistance for relay switching**: Current gain of transistor is about 110 and current drawn by relay is $75mA$. So, base current through transistor is $75/110\ mA = 0.68\ mA$. V_{BE} for the transistor is about $0.6V$. So, voltage across base resistor is nearly $4.4V$. Base resistance is $4.4/0.68 = 6.4k\Omega$. This is the upper limit on base resistance. It is because plenty of base current should flow in order to ensure that transistor is properly saturated. So, base resistance smaller than calculated should be used. Resistance of $4.7k\Omega$ or $3.3k\Omega$ is good choice. We've used base resistance of $3.3k\Omega$

2. **Series resistance for LED**: Current rating of LED is $20mA$. However, at current of $20mA$, LED glows too bright. So, we decided to reduce current to $10mA$. As forward voltage of LED is $2V$, voltage across series resistor is $3V$. So, value of series resistance is $3/10 = 0.3k\Omega$ $= 300\Omega$. Standard resistance value near to 330Ω is 330Ω. So, value of series resistance for LED was chosen to be 330Ω.

3. **Pullup Resistor**: ATMega8 has internal pullup resistor that can be activated by writing '1' to PORTX.n through program.

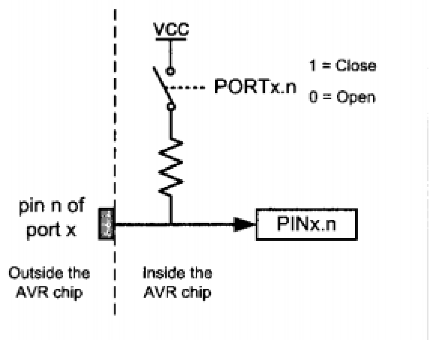


Figure 3.1: Internal pull-up resistor of ATMega8.

3.4 *Proteus* Simulation

Simplified circuit was created in *Proteus 8*. Generated hex file was loaded in ATMega8 microcontroller with proper clock source selected and simulated. Result was found to be expected. Schematic diagram for *proteus* simulation is given below.

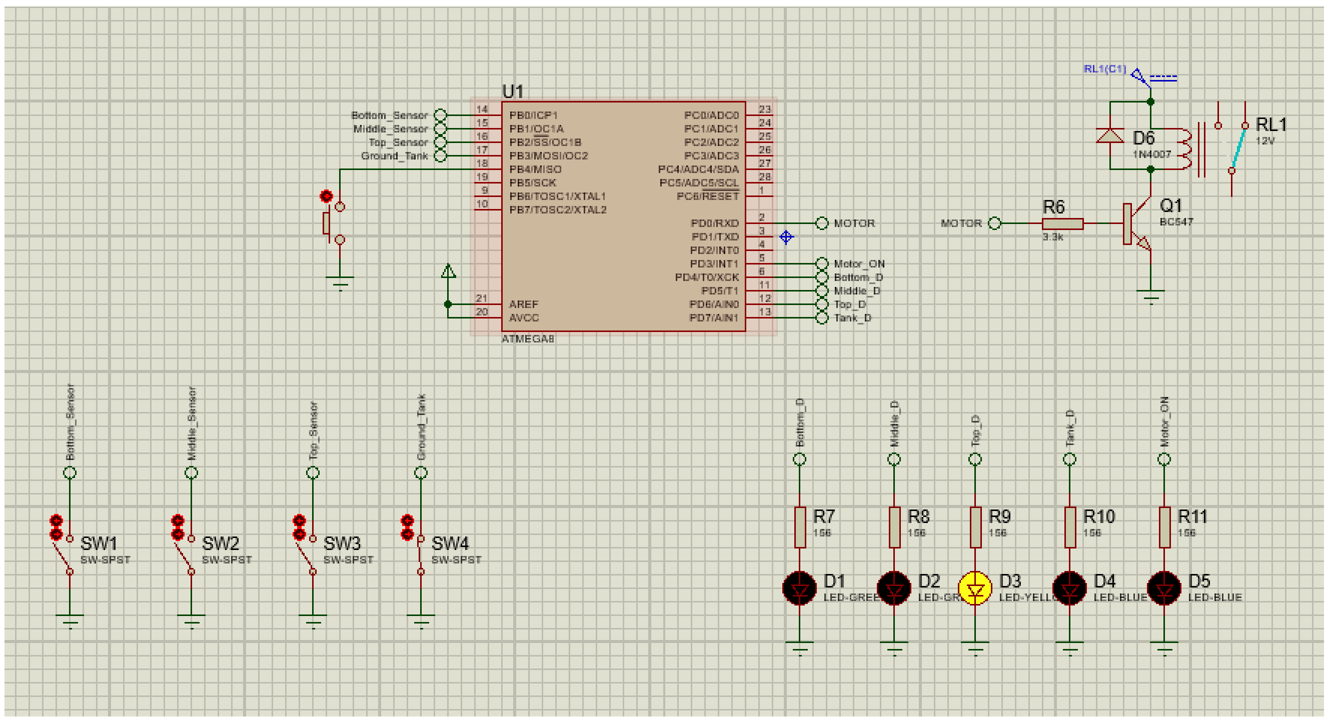


Figure 3.2: Simulation of Paani Mantra in *Proteus*.

3.5 Hardware Schematics

Complete schematic for PCB was created in *Autodesk EAGLE*. Schematic is given in next page.

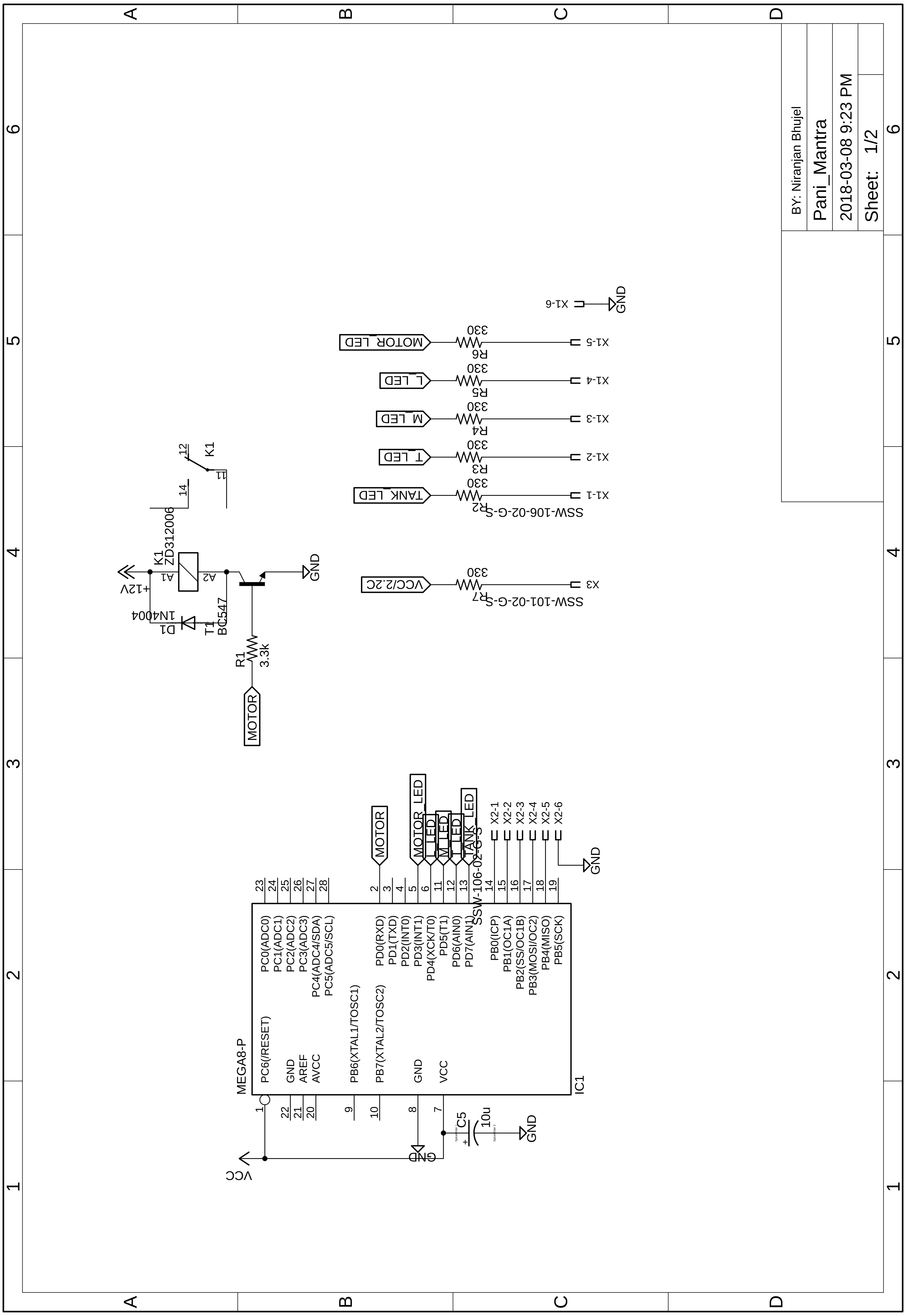


Figure 3.3: Schematic of Paani Mantra(Sheet 1).

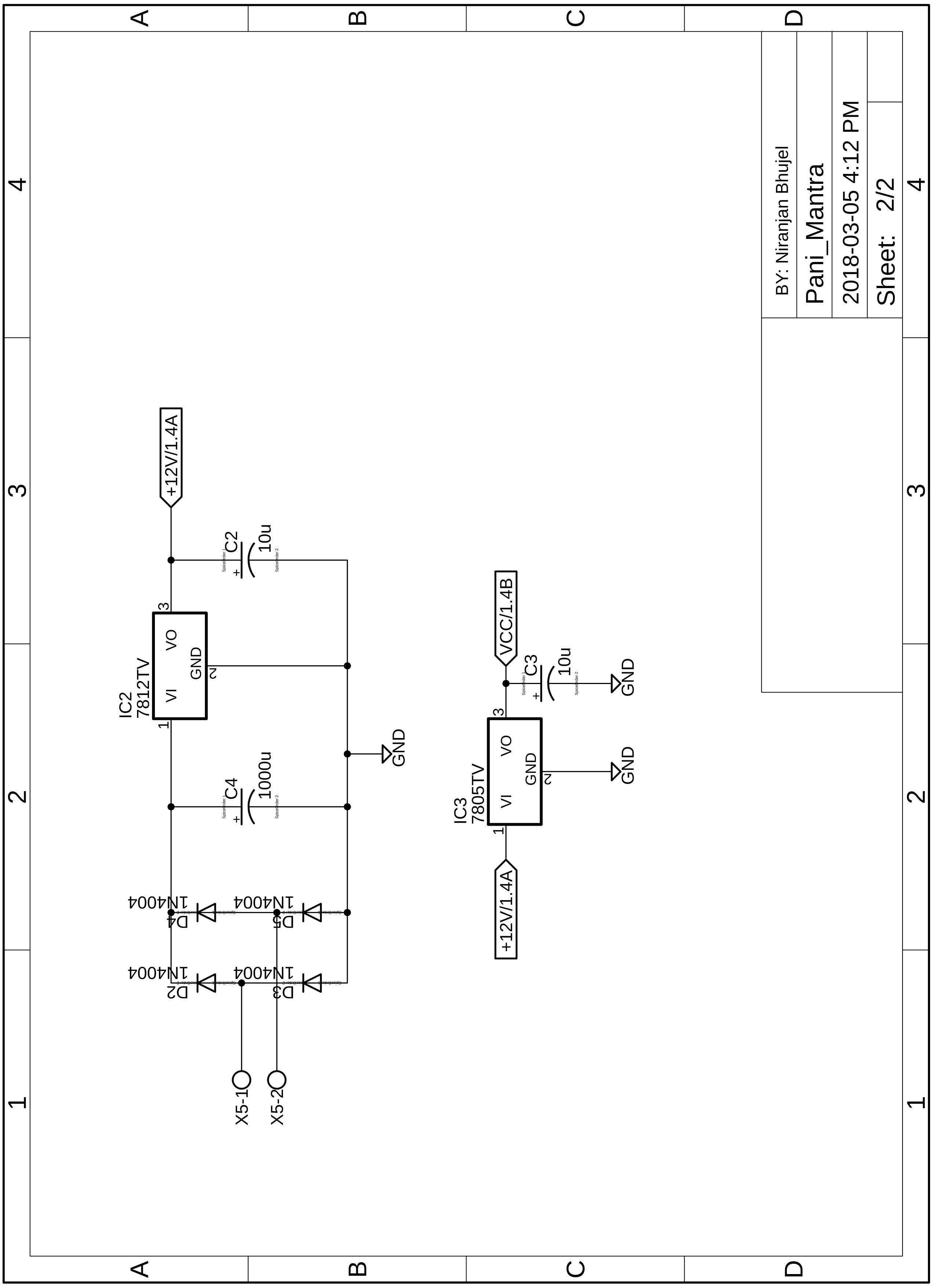


Figure 3.4: Schematic of Paani Mantra(Sheet 2).

PCB layout of Paani Mantra for external connections is given below.

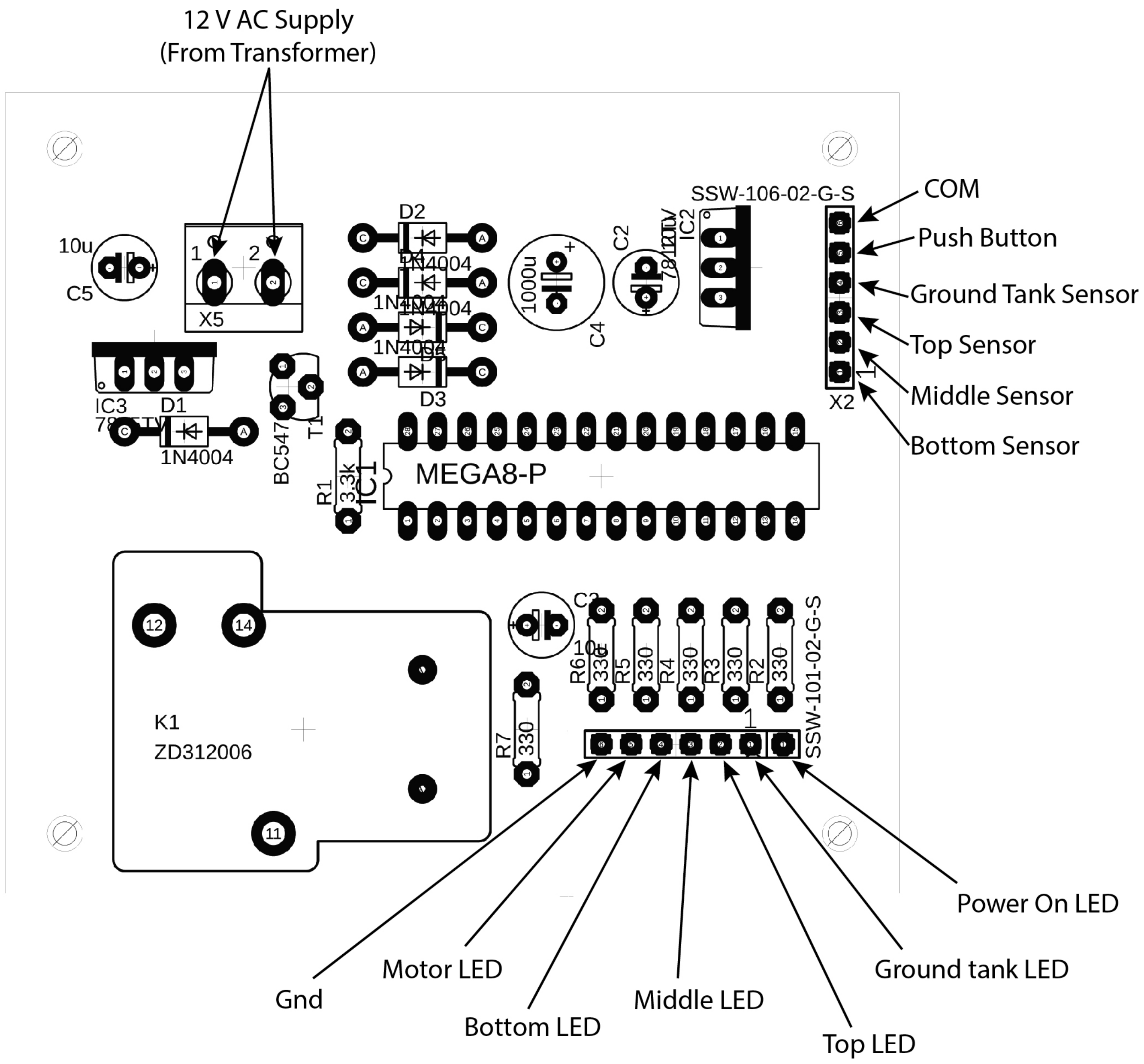


Figure 3.5: Paani Mantra PCB for external connection (viewed from component side).

Conclusion

Designed hardware was fabricated and tested internally first. Then, it was tested to control the water level of water tank at Innovative Ghar Nepal. System was left to operate for few weeks. During its operation, no flaws in design or defect in hardware was found. Also, it was found to work as expected and designed.

Index