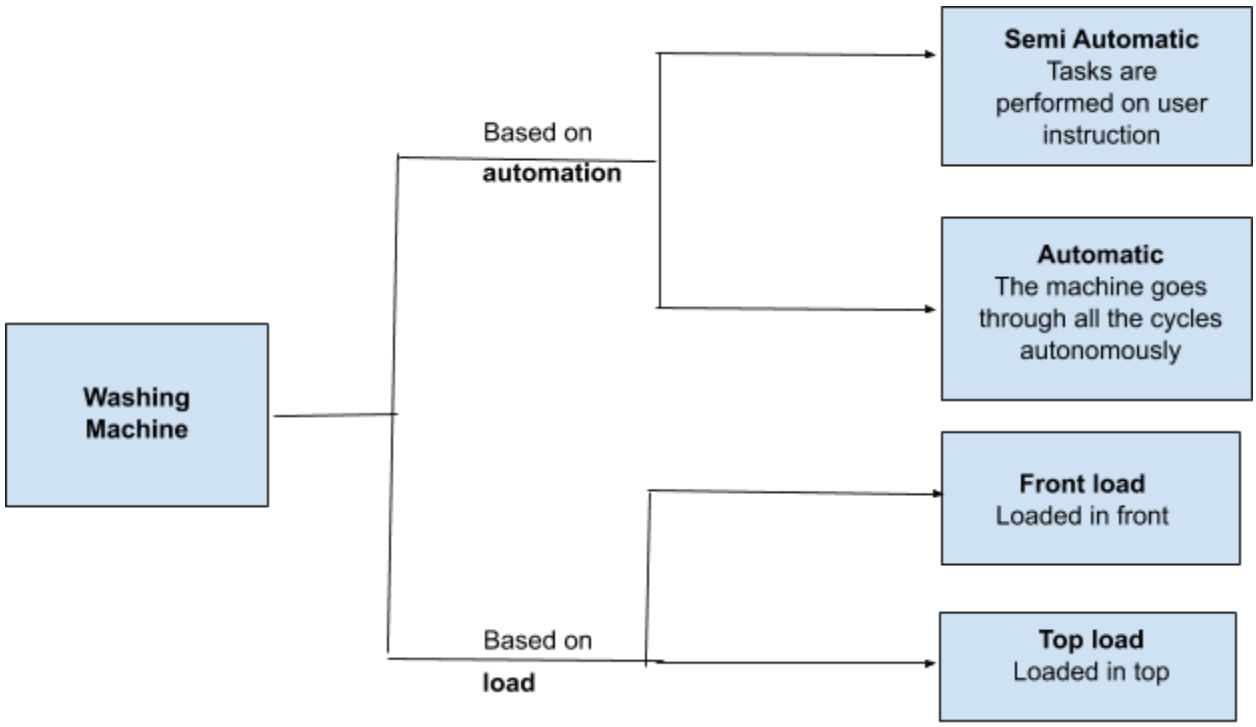
EE2016 Project Report

Washing Machine

Introduction

A washing machine is a device that automates the entire process of washing, rinsing and spinning.

Washing machines may be classified as follows.



Our Prototype Definition

While the actual functioning of a washing machine comprises of several complex stages, we have broken it down into simple functions. We have illustrated the different functions of the basic working model of a washing machine in our project using an Atmega8 processor and basic input and output devices like switches, LEDs and buzzers.

We give the user the choice to choose between three load modes: soft, medium and heavy. They can also choose the operation of washing machine: whether they want it to only wash, or only dry, or do both. Once the start button is pressed; depending on the inputs given, the outputs are displayed on the LEDs which indicate the type of process chosen (also the duration of process).

In addition, we have two external interrupts in the program, one of which stops and allows the user to reset the inputs when the button which acts as a dual start/stop is pressed during the course of the process. The other interrupt is the Lid interrupt which pauses the program when the lid is open and returns back to the process when the lid is closed back again.

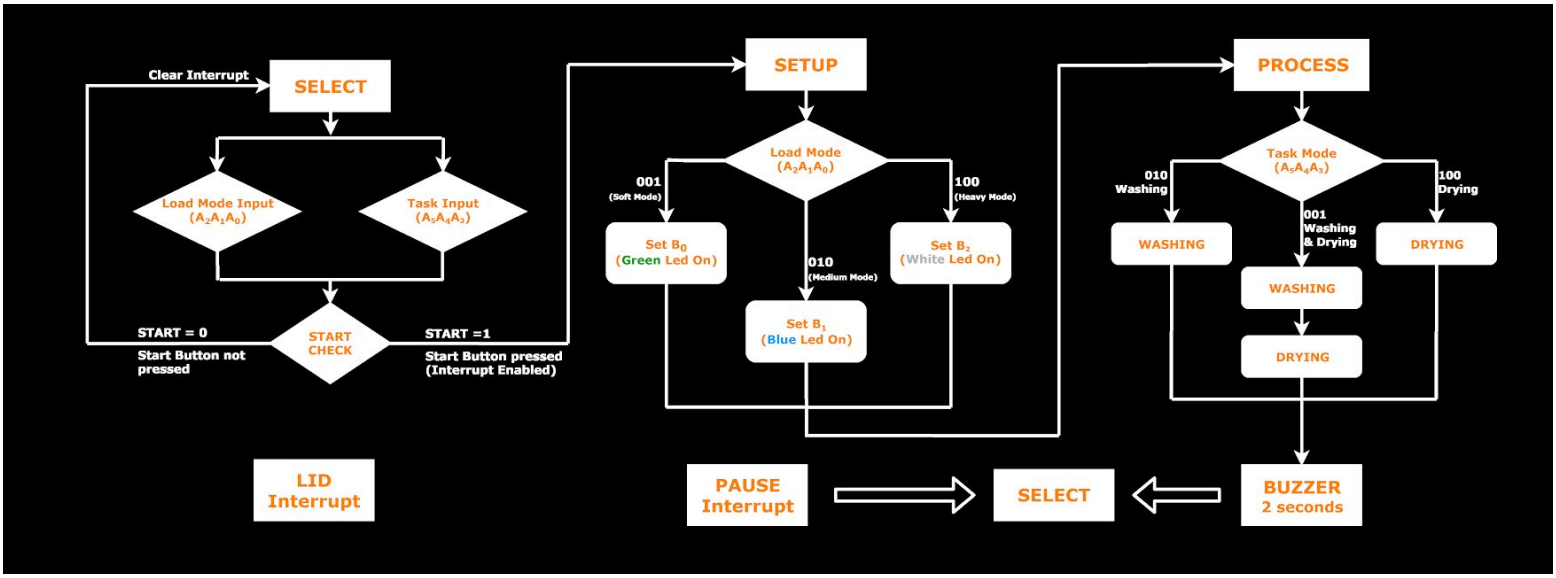
Inputs:

Soft load	Push button (PC0)
Medium load	Push button (PC1)
Heavy load	Push button (PC2)
Washing and drying	Push button (PC3)
Washing only	Push button (PC4)
Drying only	Push button (PC5)
Start/ Stop Interrupt	Push button (PD3)
Lid Interrupt	Switch (PD2)

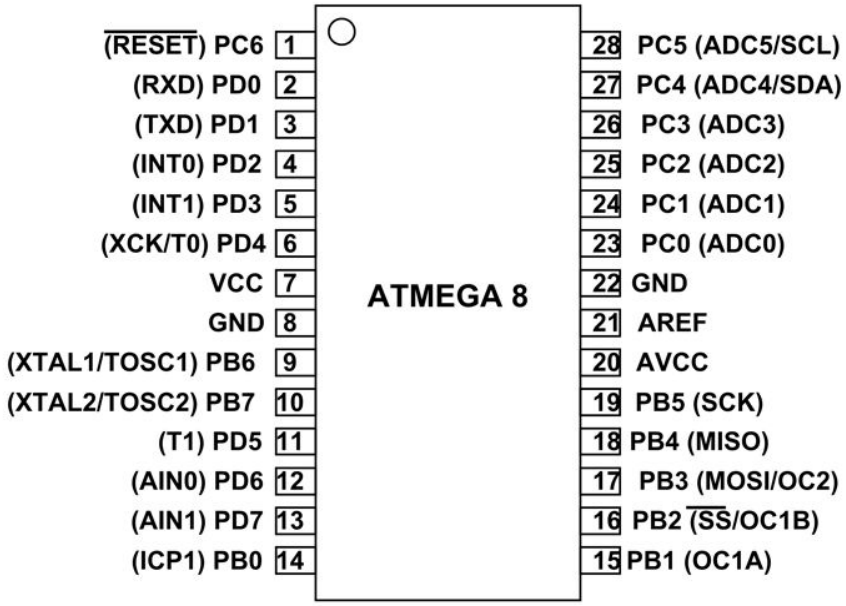
Outputs:

Soft load	Green LED glows throughout the process (PB0)
Medium load	Yellow LED glows throughout the process (PB1)
Heavy load	Blue LED glows throughout the process (PB2)
Washing with soft load	2 Red LEDs, glow thrice, with 2s delay in between (PB6)
Washing with medium load	2 Red LEDs, glow 5 times, with 2s delay in between (PB6)
Washing with heavy load	2 Red LEDs, glow 7 times, with 2s delay in between (PB6)
Drying with soft load	2 Red LEDs, glow for 4s (PB6)
Drying with medium load	2 Red LEDs, glow for 8s (PB6)
Drying with heavy load	2 Red LEDs, glow for 12s (PB6)
End of process	Buzzer rings for 2s (PB7)

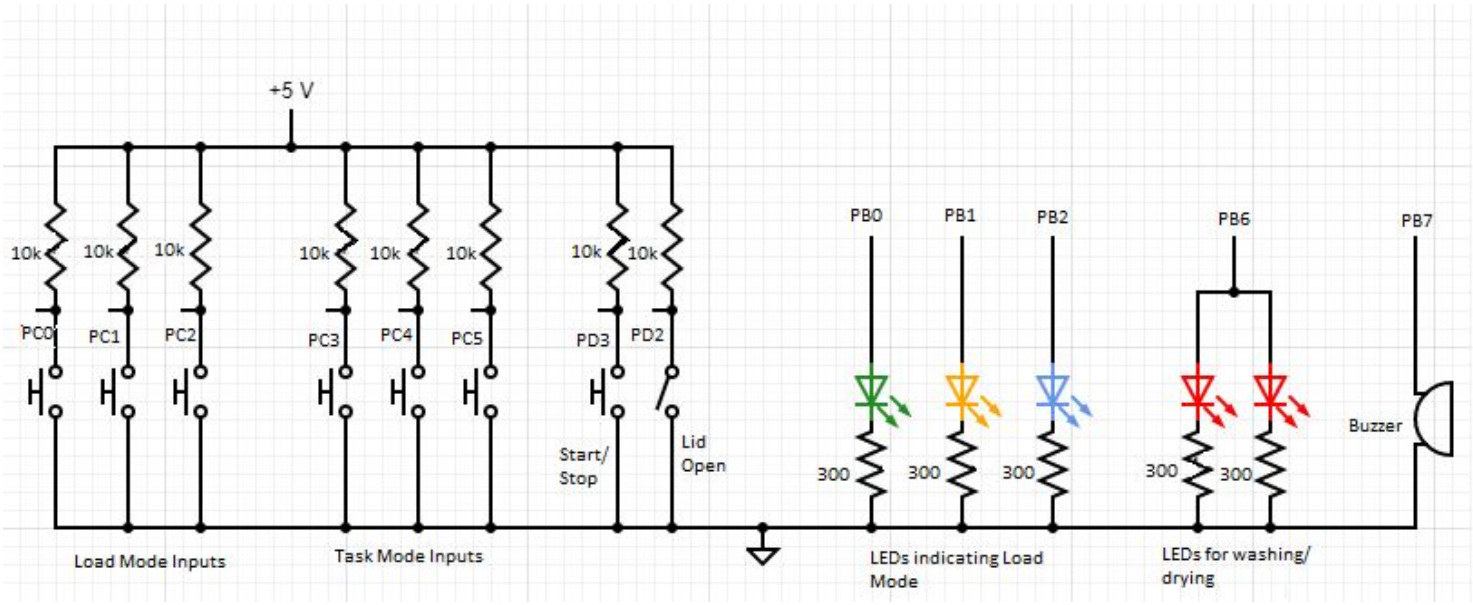
Flow Chart



Circuit Diagram:



The corresponding pins on Atmega8 are accordingly connected as shown in the circuit diagram below.



Outputs and Details

The code starts from next page -->

Code:

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

//#define F_CPU 8000000

int load= 0x01 ; // Variable for storing Load type input
int task= 0x01 ; // Variable for storing Task type input
int flag = 0; // Interrupt setting variable

void select();
void setup();
void wash(int);
void dry(int);
void blink(int);
void buzzer();

void blink(int i)
{
             // PRESCALER = 64s
 for(int k=0;k<(2*i);k++)
 {
      for(int j=0;j<250;j++)
      {
            TCCR0 = (1<<CS01)|(1<<CS00);
            TCNT0 = 0x06;
            while( (TIFR & (1<<TOV0)) == 0)
            {
                  if(flag==0){select();}
            }
            TCCR0=0;
TIFR = (1<<TOV0);
      }
 }

}

void wash(int load2)
{
 if(load2==0x04)
      {load2-=0x01;}

 for(int i=0; i<((load2*2)+1);i++)
 {
      PORTB |= (1<<PB6); // Using PB6 for red LEDs
      blink(2);
```

```
        PORTB ^= 0x40;
        blink(2);
  }
}

void dry(int load2)
{
  if(load2==0x04)
        {load2-=0x01;}

  PORTB |= (1<<PB6);
  blink(load2*4);
  PORTB ^= 0x40;
}

void buzzer()
{
  PORTB |= (1<<PB7);
  PORTB &= 0xB8;
  blink(2);
  PORTB &= 0x38 ;
}

void setup()
{
  PORTB &= 0xF8;
  PORTB |= load; // (written for PB0 PB1 PB2)

  if(task == 0x01) // Task 0x01 for wash and dry
  {
        wash(load);
        dry(load);
        buzzer();
  }
  if(task == 0x02) //Washing only
  {
        wash(load);
        buzzer();
  }
  if(task == 0x04) //Drying only
  {
        dry(load);
        buzzer();
  }

}

void select()
```

```
{
 GICR = 0x40; //Disabling INT1
 DDRB = 0b11111111;  //Green - PB0, Yellow - PB1, Blue - PB2, Red - PB6,
Buzzer - PB7
 DDRC = 0x00;    // Taking inputs LSB 6 bits
 DDRD = 0b11110011; //Interrupt inputs

 //PORTB = 0b00111000;
 //PORTC = 0b00111111;
 //PORTD = 0b00001100;

 load =0x00;task=0x00;
 PORTB &=0x38; //Ensuring that the LEDs and buzzers are 'off' before
starting the process
 while(1)
 {
if
((((~PINC)&(0x07))==0x01)||(((~PINC)&(0x07))==0x02)||(((~PINC)&(0x07))==0
x04))                                    // Taking only valid inputs
       {
            load = (~PINC)&(0x07);
       }
       if
((((~PINC)&(0x38))==0x08)||(((~PINC)&(0x38))==0x10)||(((~PINC)&(0x38))==0
x20))                                   // taking only valid inputs
       {
            task = (((~PINC)&(0x38))>>3) ; //LSB 3 bits empty in task and
now shifted
       }
       if (((~PIND)&(0x08))==0x08 && (load!=0x00) && (task!=0x00)) //
using int1 for start
       {
            flag=1;
            GICR = 0xC0; //Enabling both INT0 and INT1
            setup();        }
 }
}

ISR(INT0_vect)
{
 reti();
}

ISR(INT1_vect)
{
 flag = 0;
 GICR ^= 0x80; //disabling INT1
 reti();
```

```
}

//MAIN FUNCTION STARTS

// PD2 = INT0 and PD3= INT1

void main()
{
 GICR = 0x40; //INT0 only enabled, INT1 is disabled
 MCUCR = 0x08; //INT0: low level; INT1: falling edge

 sei();

 while (1)
 {
      select();
 }

}
```

THANK YOU