Rain Sensing Automatic Car Wiper Using STM32 Microcontroller

1. ABSTRACT

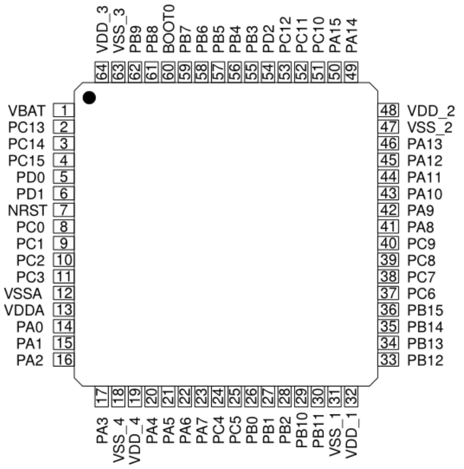
The turn of the century has seen a tremendous rise in technological advances in the field of automobiles. With 5G technology on its way and the development in the IoT sector, cars will start interacting with each other using V2V communications and become much more autonomous. In this project, an effort is made to move in the same direction by proposing a model for an automatic car wiper system that operates on sensing rain and snow on the windshield of a car. We develop a prototype for our idea by integrating a servo motor and raindrop sensor with an STM32 Microcontroller.

2. INTRODUCTION

Over the past two decades, the automotive industry has aggressively researched ways to exploit modern computing and electronic advances in the development of safety, reliability, and entertainment technologies. Despite this, automatic rain-sensing wiper systems are relatively uncommon in modern vehicles for a number of reasons. They are often too expensive, too unsightly, or too unreliable to be desired in new automobiles. Many attempts have been made at constructing an effective, reliable, and cheap rain detection and wiper control system for vehicles speed and intermittent interval automatically according to the amount of rain. To measure the amount of water usually use optical sensor. In this type of sensors uses the fact that the refraction angle and the amount of reflection of the light are different when the 2 windshield is wet. Even though optical sensors are used widely they have some disadvantage. One of disadvantages is the sensitivity to external light. Another problem is occurs when car drive at night or gone through tunnel and even in underground parking. For this many systems still activate the wiper when the car comes out of tunnels or underground parking lot. Another shortfall, maybe a major one is that the sensing area is a relatively small portion of windshield. Hence the system operate only with limited area.[2] The wiper system may fail to activate when there are some raindrops on the driver's line of sight, but not on the sensing area. They are often too expensive, too unsightly, or too unreliable to be desired in new automobiles.

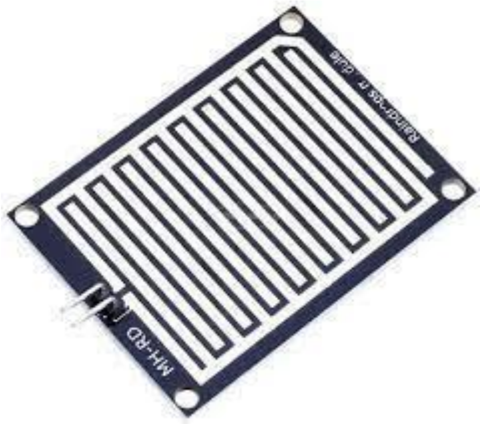
Requirements

1. STM32 Microcontroller



STM32 is **a family of 32-bit microcontroller integrated circuits by STMicroelectronics**. The STM32 chips are grouped into related series that are based around the same 32-bit ARM processor core, such as the Cortex-M33F, Cortex-M7F, Cortex-M4F, Cortex-M3, Cortex-M0+, or Cortex-M0.

2. Rain Sensor



A **rain sensor** or *rain switch* is a switching device activated by rainfall. There are two main applications for rain sensors. The first is a water conservation device connected to an automatic irrigation system that causes the system to shut down in the event of rainfall. The second is a device used to protect the interior of an automobile from rain and to support the automatic mode of window screen wipers.

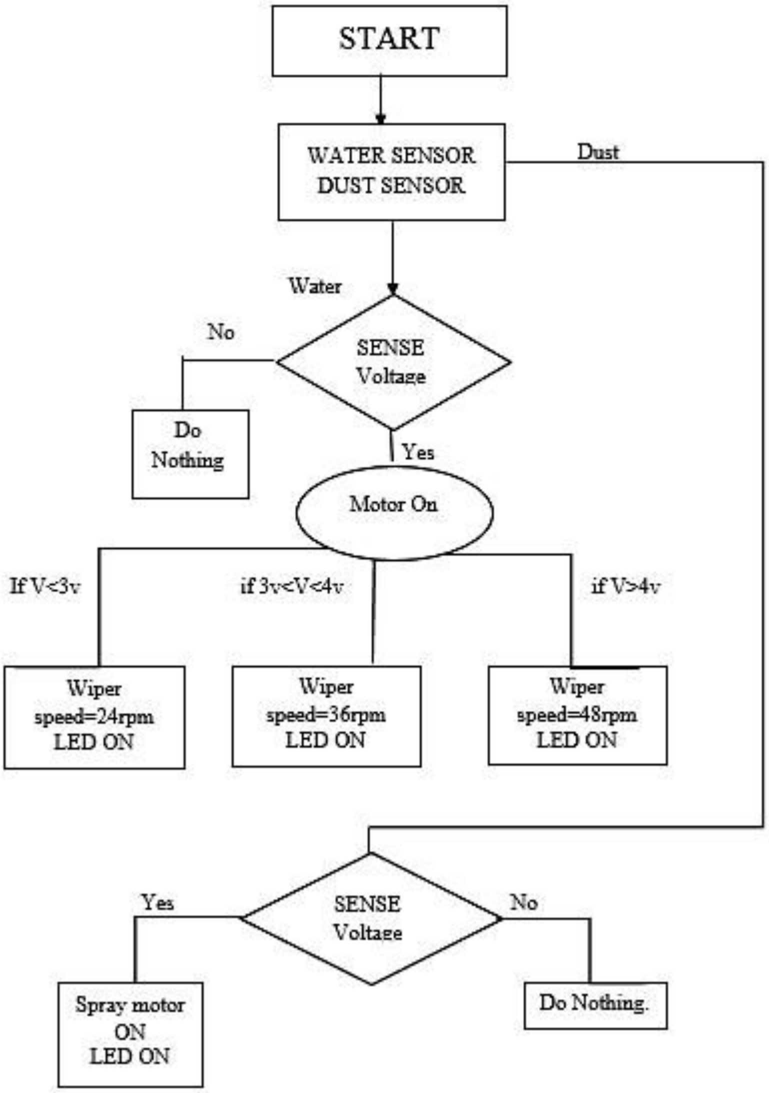
3. Servo Motor

A **servomotor** (or **servo motor**) is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity and acceleration. It consists of a suitable motocoupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors.

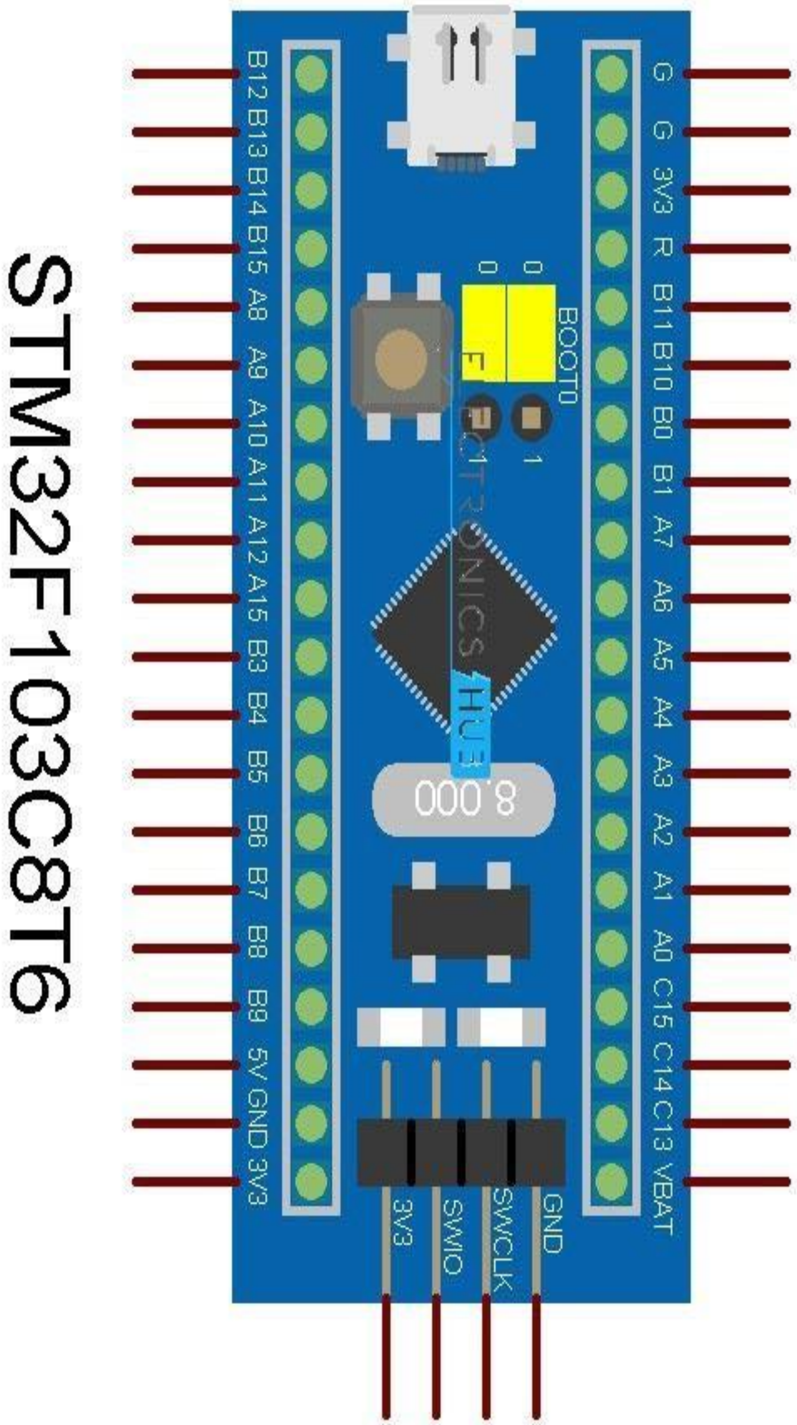
4. Wifer

Design

1. Flow Chart



2. IC pin diagram



Implementation…

1. Code

```
#include <stdint.h>
#include "stm324xx.h"
#include "gpio.h"
#include "rcc.h"

#if !defined(__SOFT_FP__) && defined(__ARM_FP)
#warning "FPU is not initialized, but the project is compiling for an FPU.
Please initialize the FPU before use."
#endif

/* RCC (Reset and clock Control) Registers */
#define RCC_BASE_ADDR              0x40023800
#define RCC_AHB1ENR_OFFSET 0x30
#define RCC_AHB1ENR                *(volatile unsigned int *)(RCC_BASE_ADDR +
RCC_AHB1ENR_OFFSET)

/* GPIO Registers */
#define GPIOD_BASE_ADDR            0x40020C00
#define GPIOD_MODER_OFFSET 0x00
#define GPIOD_MODER                *(volatile unsigned int *)(GPIOD_BASE_ADDR +
GPIOD_MODER_OFFSET)
#define GPIOD_ODR_OFFSET    0x14
#define GPIOD_ODR                  *(volatile unsigned int *)(GPIOD_BASE_ADDR +
GPIOD_ODR_OFFSET)

static volatile int flag;
static volatile int SCount = 0, count=0;        //Global variables

       //Scount is number of times the switch is pressed

       //count is simple variable to count the loop

       //Interrupt handler
       void exti0_irqhandler(void)
       {
       if(EXTI->PR & (1 << PIN_0))
       {
              flag = 1;

              EXTI->PR = EXTI->PR | (1 << PIN_0);
              }
              }

//Code to Turn on and off Red LED
long delay = 0xFFFFFFF;

       static void on_red_led(void)
       {
       volatile long i;
       GPIOD_ODR = GPIOD_ODR | 0x00001000;

       }
       static void off_red_led(void)
       {
       volatile long i;
       GPIOD_ODR = GPIOD_ODR & ~(0x00001000);
}

```

```
//GPIO Configuration
static void init_config(void)
{
config_rcc(GPIOD);
config_gpiox(GPIOD, PIN_12, GPIO_OUTPUT_PP, GPIO_SPEED_VERY_HIGH);
//Green LED

config_rcc(GPIOD);
config_gpiox(GPIOD, PIN_14, GPIO_OUTPUT_PP, GPIO_SPEED_VERY_HIGH);
//Red LED

config_rcc(GPIOD);
config_gpiox(GPIOD, PIN_13, GPIO_OUTPUT_PP, GPIO_SPEED_VERY_HIGH);
//Orange LED

config_rcc(GPIOD);
config_gpiox(GPIOD, PIN_15, GPIO_OUTPUT_PP, GPIO_SPEED_VERY_HIGH);
//Blue LED

config_rcc(GPIOA);
config_gpiox(GPIOA, PIN_0, GPIO_INPUT, GPIO_SPEED_VERY_HIGH);
//Push Button

config_gpio_irq_priority(IRQ_NO_EXTI0, NVIC_IRQ_PR_LVL_0);

//Interrupt Handler
config_gpio_interrupt(IRQ_NO_EXTI0, ENABLE); }

// Code to Turn on and off Blue LED
static void LED_1_ON()
{
/* Setting PD15 (Pin 15 of PORTD) as General Purpose Output */
GPIOD_MODER = GPIOD_MODER | 0x01000000000;

GPIOD_ODR = GPIOD_ODR | 0x00001000;
}

static void LED_1_OFF()
{
/* Setting PD15 (Pin 15 of PORTD) as General Purpose Output */
GPIOD_MODER = GPIOD_MODER | 0x01000000000;

GPIOD_ODR = GPIOD_ODR & ~0x00001000;
}
//Code to Turn on and off Green LED

static void LED_2_ON()
{
/* Setting PD12 (Pin 12 of PORTD) as General Purpose Output */
GPIOD_MODER = GPIOD_MODER | 0x01000000;

GPIOD_ODR = GPIOD_ODR | 0x00001000;
}
       static void LED_2_OFF()
       {
       /* Setting PD12 (Pin 12 of PORTD) as General Purpose Output */
       GPIOD_MODER = GPIOD_MODER | 0x01000000;

       GPIOD_ODR = GPIOD_ODR &
       ~0x00001000;
```

```
        }
    //Code to Turn on and off Orange LED
    static void LED_3_ON()
    {
    /* Setting PD13 (Pin 13 of PORTD) as General Purpose Output */
    GPIOD_MODER = GPIOD_MODER | 0x010000000;

        GPIOD_ODR = GPIOD_ODR |
        0x00001000;
        }
        static void LED_3_OFF()
        {
        /* Setting PD13 (Pin 13 of PORTD) as General Purpose Output */
        GPIOD_MODER = GPIOD_MODER | 0x010000000;

        GPIOD_ODR = GPIOD_ODR &
        ~0x00001000;
        }
    static void Delay(int x)                 // Delay function for
LED
    {
    for(int i=0;i<x;i++);    }
    //Main Function
    int main(void)
    {

        int key;
        long int A;                                          //
        variable to check for button pressed duration

        init_config();                                               //
Configuration Function for GPIO and interrupt


            //Logic for Wiper System
            while (1)
            {
            key = gpiox_read_pin(GPIOA, PIN_0);                   //
Polling

                    if(key == 1)
                    {
                    A = 0;
                    long int j;
                    for(long int j=0;j<=5000000;j++)
                    {
                    A++;
                    }

    if(A>2000000)              //1uSec = 1
    clock  hence 2 Sec = 2000000
Cycle
    {
            on_red_led();                          //Turn on Red LED

                if (flag == 1) // Set in ISR
                {
                flag = 0;
                count++;
                SCount=count % 4;
```

```
                     }


              //Wiper Moving at 1 Hz
              while(SCount==1)
              {
              LED_1_ON();
              Delay(10000);
              LED_1_OFF();
              LED_2_ON();
              Delay(10000);
              LED_2_OFF();
              LED_3_ON();
              Delay(10000);
              LED_3_OFF();
              LED_2_ON();
              Delay(10000);
              LED_2_OFF();
              LED_1_ON();
              Delay(10000);
              LED_1_OFF();
              }

              /* Wiper Moving at 4 Hz*/
              while(SCount==2)
              {
              LED_1_ON();
              Delay(2500);
              LED_1_OFF();
              LED_2_ON();
              Delay(2500);
              LED_2_OFF();
              LED_3_ON();
              Delay(2500);
              LED_3_OFF();
              LED_2_ON();
              Delay(2500);
              LED_2_OFF();
              LED_1_ON();
              Delay(2500);
              LED_1_OFF();
              }

/*Wiper Moving at 8 Hz */

                     while(SCount==3)
                     {
                     LED_1_ON();
                     Delay(1250);
                     LED_1_OFF();
                     LED_2_ON();
                     Delay(1250);
                     LED_2_OFF();
                     LED_3_ON();
                     Delay(1250);
                     LED_3_OFF();
                     LED_2_ON();
                     Delay(1250);
                     LED_2_OFF();
                     LED_1_ON();
```

```
                    Delay(1250);
                    LED_1_OFF();
                    }
                    /*Wiper Movement stop*/
                    while(SCount==0)
                    {
                    LED_1_OFF();
                    LED_2_OFF();
                    LED_3_OFF();
                    }
                    
          }

  }

      /*Wiper System turn Off when switch pressed for less than 2 sec*/
      else if(A<2000000)
                    {
                    off_red_led();
                    LED_1_OFF();
                    LED_2_OFF();
                    LED_3_OFF();
                    }
                    }
                    return 0;
     }
```