Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

REPORT ON

SPI MULTI-SLAVE SETUP FOR DUAL MOTOR SPEED CONTROL

UNDER THE GUIDANCE OF

DR. PONNAMBALAM P

SUBMITTED BY

VALLABH BHAT - 24MPE0015

PRITEE GIRASE - 24MPE0017

VIJAYKRISHNA R - 24MPE0018

NOVEMBER 2024

INDEX

S. No	Topics	Page No.
1.	Abstract	1
2.	Introduction	1
3.	Background	1- 4
4.	Requirements	5
5.	Hardware Design	5
6.	Procedure	5
7.	Results	6
8.	Conclusion	8
9.	Appendices	8-15

SPI Multi-Slave Setup for Dual Motor Speed Control

ABSTRACT

This report presents the design and implementation of an SPI-based control system for dual DC motor speed control using the ARM LPC2148 microcontroller, known for its high-performance ARM7TDMI-S core and versatile peripheral interfaces. On the transmitter side, the LPC2148's Analog to Digital Converter (ADC) is employed to enable a potentiometer for adjusting the Pulse Width Modulation (PWM), which controls the speed and direction of two distinct motors. An external interrupt, triggered via a push button on the master board, switches control between the two motors. Each motor is controlled by a slave microcontroller, with PWM signals driving chopper circuits comprising HCPL3120 opto-isolators and IRF840 MOSFETs. SPI communication is established using the serial clock (SCK), slave select (SS), MOSI, and MISO lines in a multi-slave configuration, ensuring reliable and efficient data transfer for synchronized motor operation.

Keywords: ARM, LPC2148, Pulse Width Modulation (PWM), Analog to Digital Converter (ADC).

1. INTRODUCTION

DC motor control is a cornerstone of power electronics and drives, essential for applications requiring precise and efficient speed and torque management. In industrial automation, electric vehicles, and robotics, efficient handling of motor speed directly impacts performance, energy consumption, and system reliability. This project integrates the capabilities of the LPC2148 microcontroller with advanced power electronics techniques to implement an SPI-based dual DC motor control system. Utilizing a multi-slave configuration, the system controls two distinct motors, with external interrupts triggered by a push button to switch between motors dynamically.

Efficient motor speed handling is achieved by dynamically adjusting the PWM pulse width through the LPC2148's ADC, using a potentiometer on the transmitter side. This allows for fine-grained control over motor speed and direction, catering to varying load conditions and operational requirements. The PWM signals are processed by individual chopper circuits, incorporating HCPL3120 opto-isolators for electrical isolation and IRF840 MOSFETs for high-speed switching. This setup ensures minimal power loss, improved motor efficiency, and enhanced system safety.

SPI communication, established through SCK, SS, MOSI, and MISO lines, ensures reliable and fast data transfer between the master and the two slave microcontrollers. The ability to efficiently manage motor speed across two independent motors not only optimizes energy usage but also extends motor lifespan, showcasing the synergy between embedded systems and power electronics in modern drive applications.

2. BACKGROUND

The Serial Peripheral Interface (SPI) is a synchronous communication protocol widely used for short-distance communication in embedded systems. It operates in a master-slave configuration, where the master device initiates and controls communication, while the slave devices respond to the master's commands.

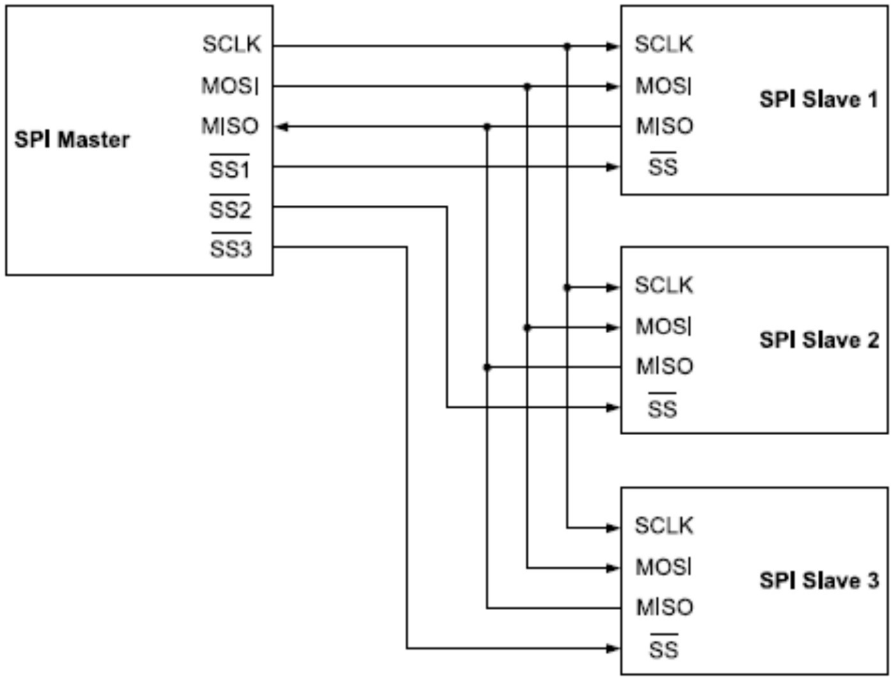

Figure 1 SPI Master Connected to Three SPI Slaves

SPI uses four key signals:

- *SCK (Serial Clock):* This is a serial clock signal that is transmitted by the bus master to the slave device(s).
- *MOSI (Master Out Slave In):* The data from a master to a slave device, is transmitted using Master-Out-Slave-In (MOSI) data line.
- *MISO (Master In Slave Out):* The data to a master device from a slave, is transmitted using Master-In-Slave-Out (MISO) data line.
- *SS (Slave Select):* This signal is used to select or enable a slave device for SPI communication. This signal is active low i.e., a logic low on this line enables the device for SPI communication.

In operation, data is shifted out of the master and into the slave, and vice versa, on the rising or falling edges of the clock signal. This full-duplex communication makes SPI highly efficient for real-time control applications. SPI supports high data transfer rates, making it ideal for quick updates to control signals, which is essential for dynamic motor speed and direction adjustments. Its straightforward structure simplifies the integration of microcontrollers with motor driver circuits, ensuring efficient communication and control.

2.1. LPC2148 MICROCONTROLLER

The LPC2148 microcontroller, based on the ARM7TDMI-S core, is widely used in embedded applications for its computational efficiency and peripheral support.

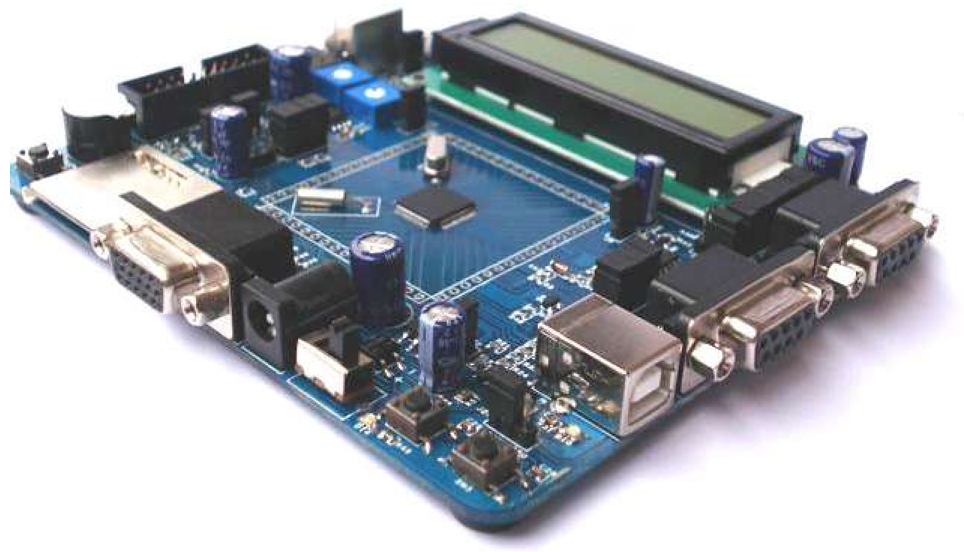

Figure 2 NGX ARM7 LPC2148 Development Board

Key features relevant to this project include:

- *PWM (Pulse Width Modulation):* The LPC2148 provides multiple PWM channels, enabling precise control over motor speed and direction by modulating the duty cycle of the signal.
- *ADC (Analog-to-Digital Converter):* The built-in ADC modules allow real-time conversion of analog inputs, such as the potentiometer position, to digital signals for processing.
- *SPI Interface:* Supports efficient communication with external devices like motor drivers, ensuring seamless integration in control systems.

2.2. DC MOTOR CONTROL

A DC motor converts electrical energy into mechanical motion, with speed proportional to the applied voltage. By adjusting the voltage or current, the motor's speed and direction can be precisely controlled. PWM is a common method used to achieve this, where the average voltage applied to the motor is controlled by varying the duty cycle of a digital pulse.

In this project, the PWM signals are generated using the LPC2148 microcontroller and transmitted via SPI to control a motor driver circuit. The motor driver circuit includes:

HCPL3120 Opto-Isolator: Provides electrical isolation between the control circuit and the high-power motor, ensuring safety and noise immunity.

IRF840 MOSFET: Acts as a high-speed switching device in the chopper circuit, efficiently driving the DC motor.

The SPI protocol facilitates smooth and reliable data transfer between the microcontroller and the driver circuit, enabling real-time adjustments to motor speed and direction. This integration demonstrates the effectiveness of SPI in implementing efficient and responsive motor control systems.

3. REQUIREMENTS

Component	Quantity
LPC2148 Boards	3
IRF840 MOSFET	2
HCPL3120	2
DC Motor	2
Bread board	3
Regulated Power Supply	2
Connecting Wires	As Reqd

4. HARDWARE DESIGN

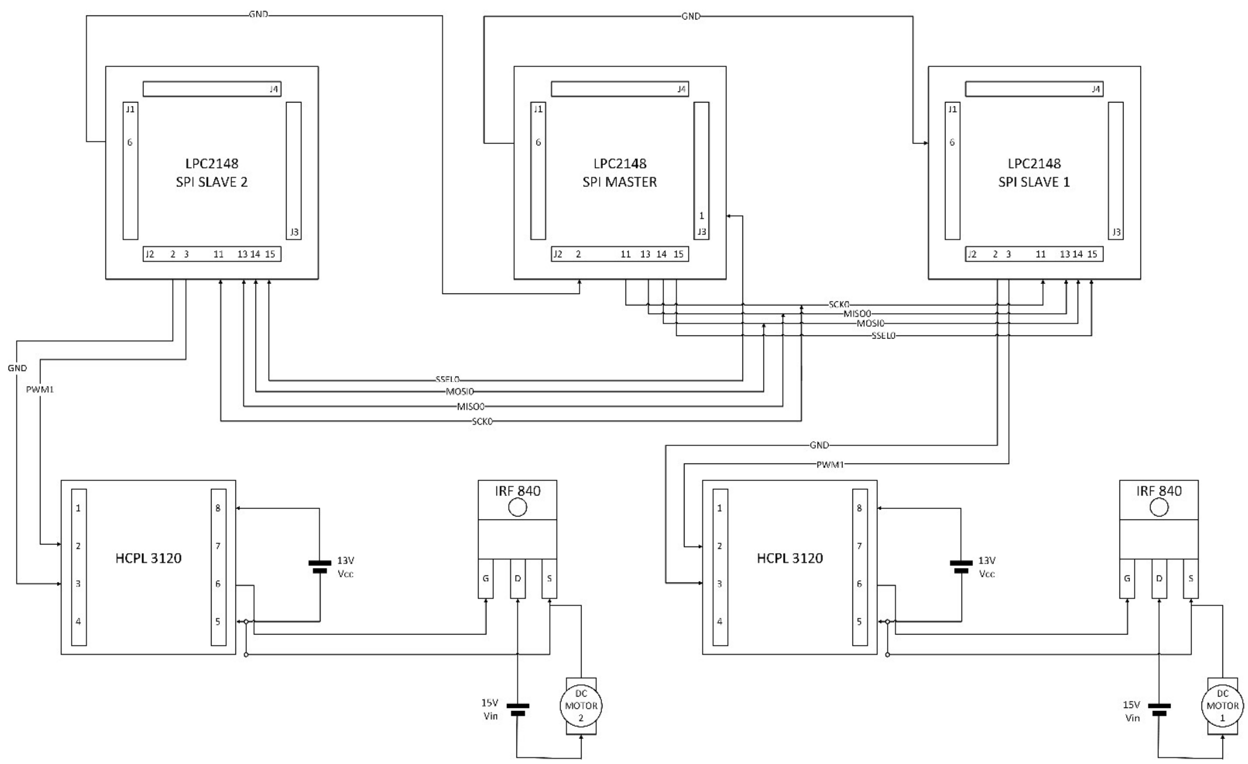

Figure 3 Connection Configurations

5. PROCEDURE

- Place two LPC2148 microcontrollers on separate development boards, designating one as the SPI Master and the other as the SPI Slave.

- Connect SCK0 (Serial Clock) PIN 11 of the master to the breadboard and then take it to PIN 11 of both the slaves.
- Connect MISO (Master In Slave Out) PIN 13 of the master to the breadboard and then take it to PIN 13 of both the slaves.
- Connect MOSI (Master Out Slave In) PIN 14 of the master to the breadboard and then take it to PIN 14 of both the slaves.
- Connect SSEL0 (Slave Select) PIN 15 of the master to PIN 15 of the slave 1 and PIN 1 from J3 Junction of the master to PIN 15 of slave 2.
- Ensure a common ground is established between the master and slave microcontrollers.
- Initialize the ADC module on the master to read the potentiometer's analog voltage.
- Configure the PWM module to output the signal to the HCPL3120 opto-isolator as displayed in Figure 3
- Power the HCPL3120 opto-isolator with a 13V supply and connect its output to the gate of the IRF840 MOSFET for both the chopper circuits.
- Wire the MOSFET in a chopper configuration, with the drain connected to the DC motor and the source to ground.
- Supply 15V to the motor through the MOSFET's drain terminal.
- Verify that the chopper circuit modulates the motor's speed according to the PWM signal.

6. RESULTS

6.1. HARDWARE IMPLEMENTATION

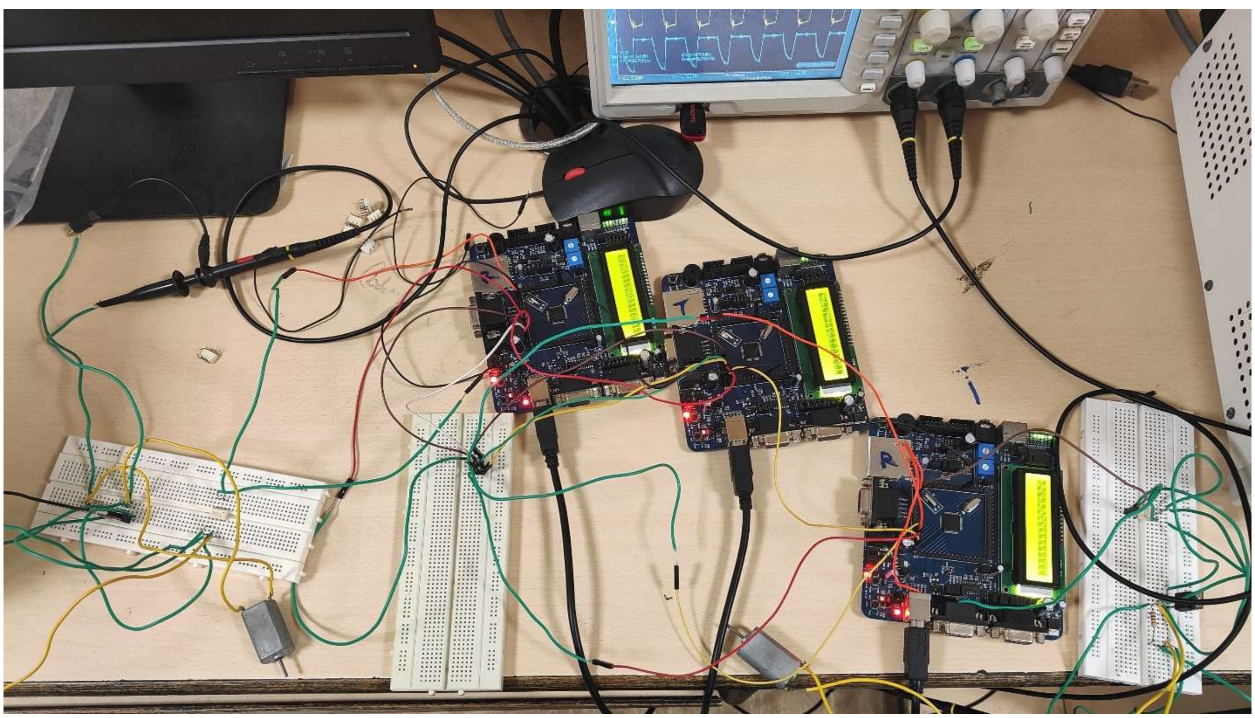

Figure 4 Board Connected to Chopper Circuit

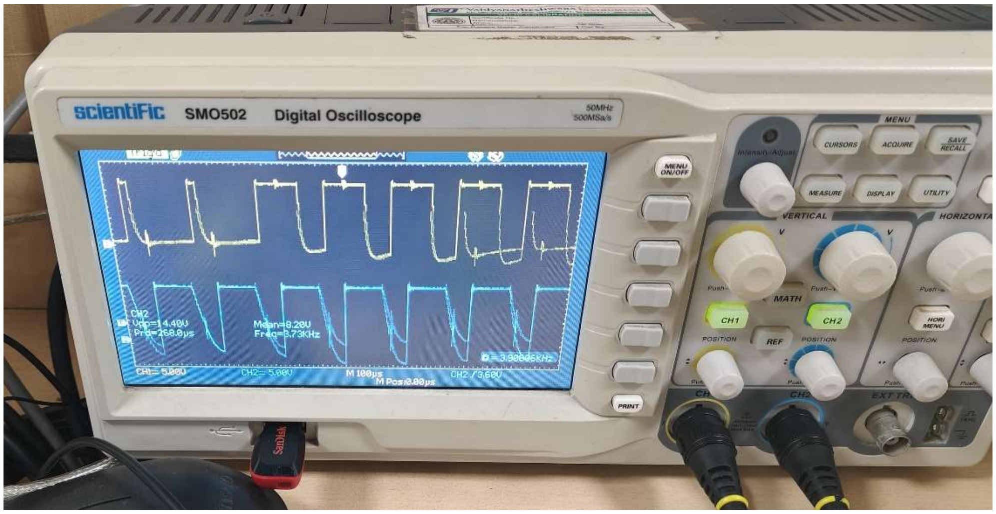

Figure 5 DSO Output Waveforms

6.2. OBSERVED WAVEFORMS

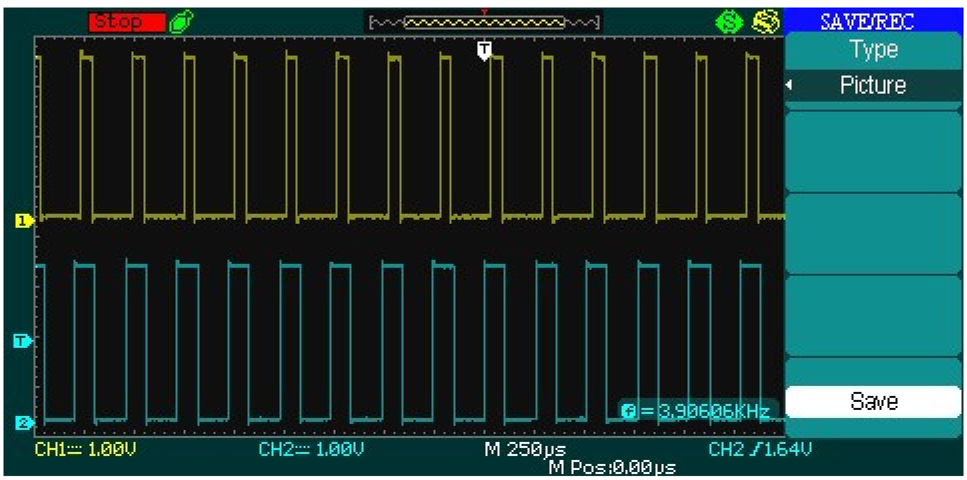

Figure 6 Output Observed Across the Slave Board

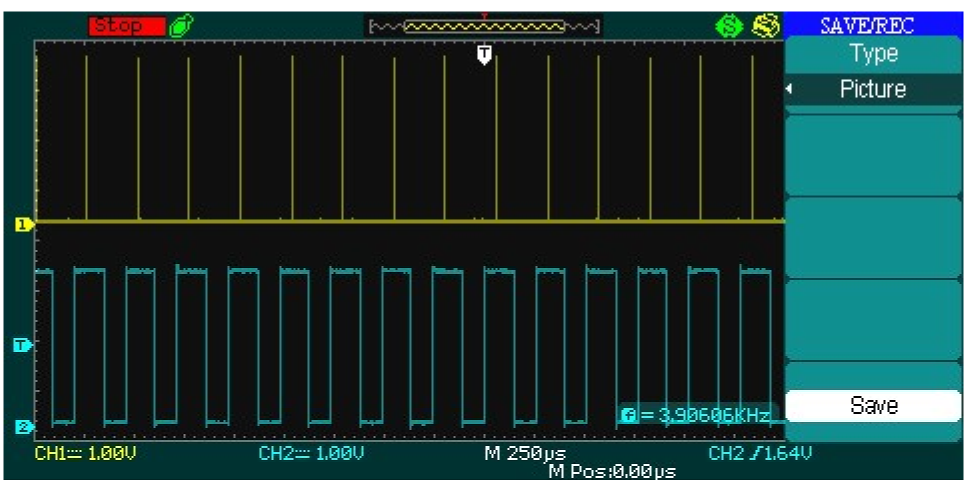

Figure 7 Output Observed Across the Load

7. CONCLUSION

The implementation of an SPI-based dual DC motor control system using LPC2148 microcontrollers demonstrates an efficient and reliable approach to managing the speed and direction of multiple motors. By leveraging the multi-slave configuration of SPI, the system ensures seamless and high-speed communication between the master and two slave modules. The integration of ADC for potentiometer input on the transmitter side, along with external interrupts to dynamically switch control between motors, provides real-time and flexible motor speed management. Each motor is controlled via a dedicated PWM signal processed by a robust chopper circuit, ensuring high efficiency and adaptability to varying load conditions.

This setup underscores the potential of SPI protocol and embedded systems in managing complex motor control tasks in power electronics and drives applications. The project paves the way for scalable, multi-motor control solutions that are well-suited for industrial automation, robotics, and other advanced mechatronics systems, where precision, efficiency, and reliability are paramount.

8. APPENDICES

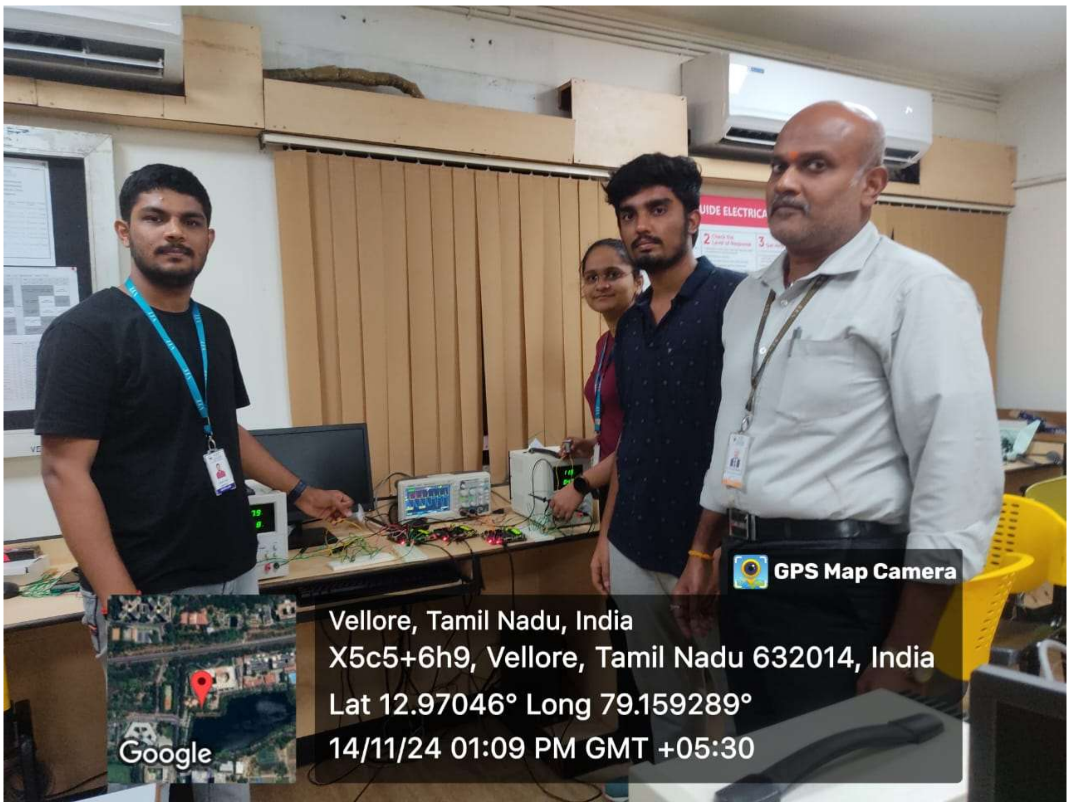

Figure 8 Contributors of the Work

8.1. SPI TRANSMITTER CODE

```
area SPITransmit, code, readonly
S0SPCR EQU 0xE0020000
S0SPSR EQU 0xE0020004
```

```
S0SPDR EQU 0xE0020008
S0SPCCR EQU 0xE002000C
S0SPINT EQU 0xE002001C
PINSEL0 EQU 0xE002C000
IO0DIR EQU 0XE0028008
IO0SET EQU 0XE0028004
IO0CLR EQU 0XE002800C
IO0PIN EQU 0XE0028000
PINSEL1 EQU 0XE002C004
AD0CR EQU 0XE0034000
AD0DR3 EQU 0XE003401C
AD0GDR EQU 0XE0034004
AD0INTEN EQU 0XE003400C
AD0STAT EQU 0XE0034030
VICIntEnable EQU 0XFFFFF010
VICIntSelect EQU 0XFFFFF00C
VICVectCntl0 EQU 0XFFFFF200
VICVectAddr0 EQU 0XFFFFF100
VICVectAddr EQU 0XFFFFF030
VICVectCntl1 EQU 0XFFFFF204
VICVectAddr1 EQU 0XFFFFF104
EXTINT EQU 0XE01FC140
        LDR R0, = PINSEL0  ; Select P0.4, P0.5, P0.6, P0.7 as SCK0, MISO0, MOSI0 and GPIO and P0.15 as EINT2
        LDR R1, = 0X80001500
        STR R1,[R0]
        LDR R1, =PINSEL1 ; Address Of PINSEL1
        LDR R2, =0X10000000 ; Activating AD0.3 at P0.30
        STR R2, [R1]
        LDR R2, =VICINTENABLE
        MOV R1, #0X50000
        STR R1,[R2]
```

```
LDR R2, =VICIntSelect
MOV R1, #0X0
STR R1,[R2]
LDR R2, =VICVectCntl0
MOV R1, #0X32
STR R1,[R2]
LDR R2, =VICVectAddr0
LDR R1, =IRQ_Handler
STR R1,[R2]
LDR R2, =VICVectCntl1
MOV R1, #0X30
STR R1,[R2]
LDR R2, =VICVectAddr1
LDR R1, =IRQ_Handler1
STR R1,[R2]
LDR R1, =AD0INTEN
LDR R2, =0X08
STR R2, [R1]
LDR R1, =AD0CR ; Address Of AD0 Control Register
LDR R2, =0X00251508 ; 2 is to make a/d converter operational from power down mode, 5 is for setting bust mode and making the adc to go to 8 bit mode f is for setting clk/div and 8 is for selecting channel 3.
str R2, [R1]
LDR R0, = S0SPCR ; SPI Master mode, 8-bit data, SPI0 mode
LDR R1, = 0X0030
STR R1,[R0]
LDR R0, = S0SPCCR ; Even number, minimum value 8, pre scalar for SPI Clock
LDR R1, = 0X0010
STR R1,[R0]
LDR R0, = IO0DIR
LDR R1, = 0X180
STR R1,[R0]
```

```
        MOV R5,#0X0100
        MOV R4,#0X080
L1      B L1
IRQ_Handler1
        SUB LR,LR,#4
        MOV R6, LR
        MOV R5,#0X080
        MOV R4,#0X0100
        LDR R2, =VICVectAddr
        LDR R1, =0x0
        STR R1, [R2]
        LDR R2, =EXTINT
        LDR r1, =0x04
        STR R1, [R2]
        MSR     CPSR_c, #Mode_USR
        MOV LR, R6
        BX LR
IRQ_Handler
        SUB LR,LR,#4
        MOV R6, LR
        LDR R3, =AD0DR3 ; Address 0f AD0 Global Data Register
        LDR R2, [R3]
        LDR R1, =0X0000FF00
        AND R7,R1,R2
        LSR R8,R7,#8
        LDR R11, =0X40000000
        STR R8,[R11]
        LDR R0, = IO0PIN
        STR R5,[R0]
        LDR R0, = S0SPDR
        STR R8,[R0]
```

```
L       LDR R0, = S0SPSR
        LDR R1,[R0]
        LDR R2,=0x80
        AND R3,R2,R1
        CMP R3,R2
        BNE L
        LDR R0, = IO0SET
        MOV R1, R4
        STR R1,[R0]
        LDR R2, =VICVectAddr
        LDR R1, =0x0
        STR R1,[R2]
        MSR     CPSR_c, #Mode_USR
        MOV LR, R6
        BX LR
        END
```

8.2. SPI RECEIVER CODE

```
        area SPIReceive, code, readonly
S0SPCR EQU 0xE0020000
S0SPSR EQU 0xE0020004
S0SPDR EQU 0xE0020008
S0SPCCR EQU 0xE002000C
S0SPINT EQU 0xE002001C
PINSEL0 EQU 0xE002C000
IO0DIR EQU 0XE0028008
IO0SET EQU 0XE0028004
IO0CLR EQU 0XE002800C
IO0PIN EQU 0XE0028000
PWMMR1 EQU 0XE001401C
PWMLER EQU 0XE0014050
```

```
PWMPCR EQU 0XE001404C
PWMPR EQU 0XE001400C
PWMMR0 EQU 0XE0014018
PWMMCR EQU 0XE0014014
PWMTCR EQU 0XE0014004
VICIntEnable EQU 0XFFFFF010
VICIntSelect EQU 0XFFFFF00C
VICVectCntl0 EQU 0XFFFFF200
VICVectAddr0 EQU 0XFFFFF100
VICVectAddr EQU 0XFFFFF030
S0SPINT EQU 0xE002001C
        LDR R0, = PINSEL0  ; Select P0.4, P0.5, P0.6, P0.7 as SCK0, MISO0, MOSI0 and GPIO and P0.0 as PWM1
        LDR R1, = 0X05502
        STR R1,[R0]
        LDR R0, = S0SPCR ; SPI Slave mode, 8-bit data, SPI0 mode
        LDR R1, = 0X0080
        STR R1,[R0]
        LDR R0, = S0SPCCR ; Even number, minimum value 8, pre scalar for SPI Clock
        LDR R1, = 0X0090
        STR R1,[R0]
        LDR R2, =VICIntEnable ; activating the SPIO interrupt
        MOV R1, #0X400
        STR R1,[R2]
        LDR R2, =VICIntSelect
        MOV R1, #0X0
        STR R1,[R2]
        LDR R2, =VICVectCntl0
        MOV R1, #0X2A
        STR R1,[R2]
        LDR R2, =VICVectAddr0
        LDR R1, =IRQ_Handler
```

```
        STR R1,[R2]
        LDR R0, = IO0DIR
        LDR R1, = 0X0FFFFFFFF
        STR R1,[R0]
        LDR R0, =PWMPR
        LDR R1, =0X02
        STR R1,[R0]
        LDR R0, =PWMMR0
        LDR R1, =0X100
        STR R1,[R0]
        LDR R0, =PWMMR1
        MOV R1, #0X01
        STR R1,[R0]
        LDR R0, =PWMMCR
        MOV R1, #0X02 ; makes the 1st bit of PWMMCR register to go high which resets the PWMTC when PWMMR0 matches
        STR R1,[R0]
        LDR R0, =PWMLER
        MOV R1, #0X03 ; makes the 0th and 1st bit of PWMLER to go high which activates PWMMR0 and PWMMR1 register contents
        STR R1,[R0]
        LDR R0, =PWMPCR
        MOV R1, #0X0200 ; makes the 9th bit of PWMPCR register to go high which enables the PWM1
        STR R1,[R0]
        LDR R0, =PWMTCR
        MOV R1, #0X02 ; makes the 1st bit of PWMTCR register to go high which resets the PWM timer counter
        STR R1,[R0]
        LDR R0, =PWMTCR
        MOV R1, #0X09 ; makes the 3rd and 0th bit of PWMTCR register to go high which makes the counter enable (0th bit) and PWM Enable (3rd bit)
        STR R1,[R0]
L1      B L1
```

```
IRQ_Handler
        SUB LR,LR,#4
        MOV R6, LR
L       LDR R0, = S0SPSR
        LDR R1,[R0]
        LDR R2,=0x80
        AND R3,R2,R1
        CMP R3,R2
        BNE L
        LDR R0, = S0SPDR
        LDR R8,[R0]
        LDR R0, =PWMMR1
        STR R8,[R0]
        LDR R0, = IO0PIN
        STR R8,[R0]
        LDR R2, =VICVectAddr
        LDR R1, =0x0
        STR R1,[R2]
        LDR R2, =S0SPINT
        MOV R1, #0X01
        STR R1,[R2]
        LDR R0, =PWMLER
        MOV R1, #0X03 ; makes the 0th and 1st bit of PWMLER to go high which activates PWMMR0 and PWMMR1 register contents
        STR R1,[R0]
        MSR     CPSR_c, #Mode_USR
        MOV LR, R6
        BX LR
        END
```