**Design of Speedometer Interface**

**using MSP430 and Hall Sensor**

**Course:** Embedded C Special Topic

**Team Names:**

Gaurav Sai Palasari - PES1UG21EE030

Siddharth D Srinivas - PES1UG21EC283

Srividya Prasad - PES1UG21EC297

**Table of Contents:**

Contents

[Objective 3](#_Toc141707369)

[Control flow diagram 3](#_Toc141707370)

[Circuit 4](#_Toc141707371)

[Code Decomposition 4](#_Toc141707372)

[Conclusion 6](#_Toc141707373)

[Results 6](#_Toc141707374)

# Objective

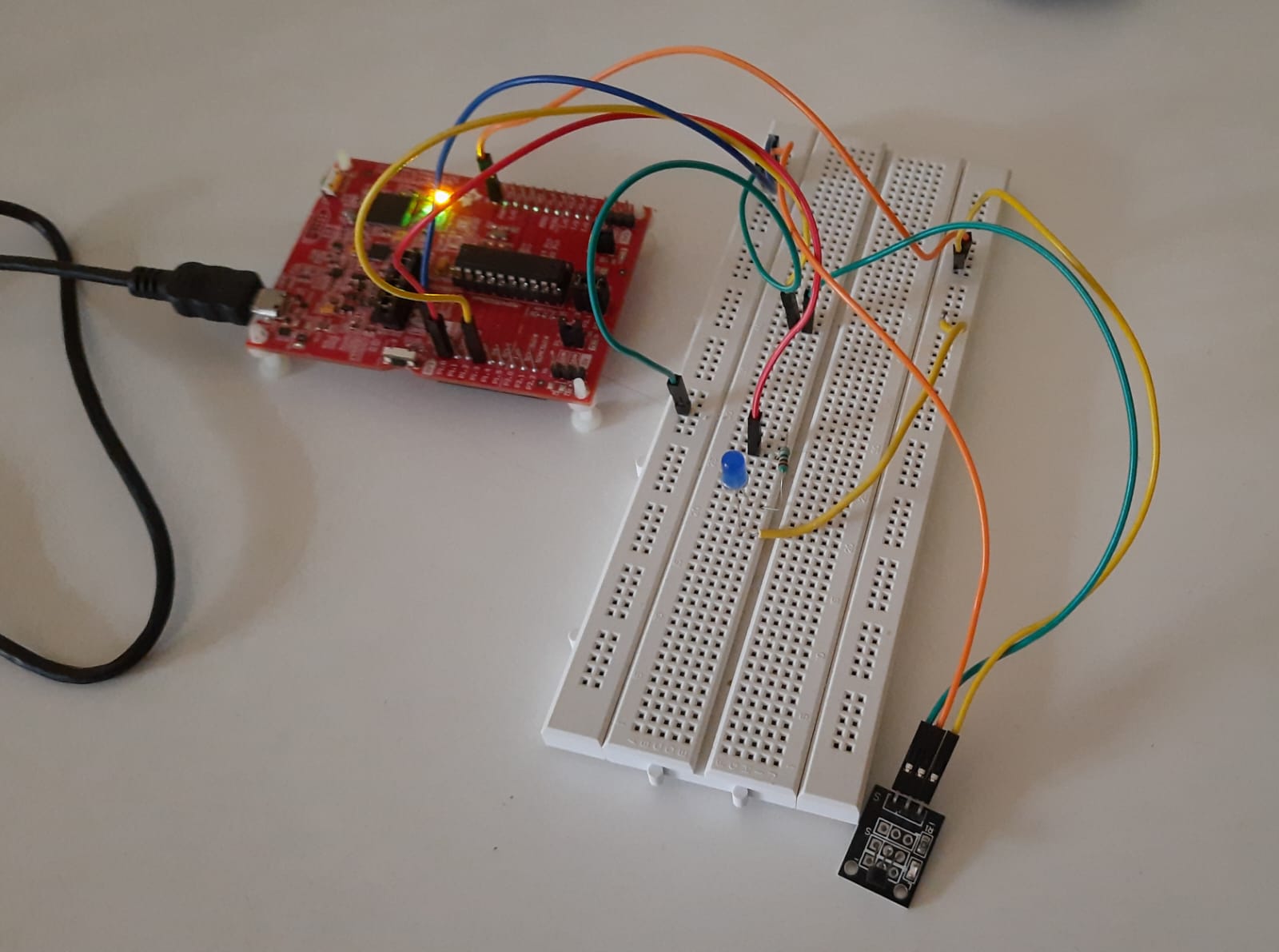
The RPM Measurement System using MSP430 is a robust and accurate solution designed to measure the rotational speed of rotating objects, such as wheels, using a Hall effect sensor. The design and implementation of the system, modular architecture and key functionalities are discussed as below.

The purpose of this modularized code is to provide an application that can measure RPM, Pulse per second, and Speed using a Hall Sensor and also blink an LED when required. The code is divided into three modules: HAL (Hardware Abstraction Layer), DRV (Hardware Driver Layer), and the main application.

# Control flow diagram

Link: <https://drive.google.com/file/d/1_VnPdndJS8WpIyg9Ratq0rTvKH1edjGV/view?usp=sharing>

# Circuit



# Code Decomposition

1. main.c: Main Application

The main application is where the user can invoke the services provided by the HAL and DRV layers. It configures the LED and Hall Sensor, sets the wheel diameter, and initializes the necessary components. The main application also handles the Hall Sensor interrupt and periodically calculates RPM, PPS, and speed.

1. hal.c/hal.h: Hardware Abstraction Layer

The HAL module acts as an interface between the Hardware Driver Layer (DRV) and the main application. It provides functions to initialize the LED and Hall Sensor, set the wheel diameter, reset pulse count, calculate RPM, PPS, and speed, and manage the Hall Sensor interrupt callback. This module interacts with the DRV layer to control the hardware.

1. drv.c/drv.h: Hardware Driver Layer

The DRV module directly interacts with the hardware to configure the LED and Hall Sensor pins as input/output. It defines the LED blinking function and sets up the Hall Sensor interrupt service routine. The DRV layer is tightly coupled with the hardware and provides low-level control.

Functions:

**HAL (Hardware Abstraction Layer)**

void hal\_setWheelDia(float wd): Sets the wheel diameter for speed calculation.

void hal\_ledInit(): Initializes the LED pin as an output.

void hal\_hallInit(): Initializes the Hall Sensor pin as an input with interrupt.

void resetPulseCount(): Resets the pulse count to zero.

UINT16 getPPS(): Calculates and returns pulses per second.

UINT16 getRPM(): Calculates and returns revolutions per minute.

FLOAT32 getSpeed\_mps(): Calculates and returns speed in meters per second.

void printPPS(): Prints pulses per second.

void printRPM(): Prints revolutions per minute.

void printSpeed\_mps(): Prints speed in meters per second.

void hal\_blinkLED(): Blinks the LED.

void incPulseCount(): Increments the pulse count.

void hal\_process\_HallSensor(): Starts interval measurement and waits for a specified interval.

void hal\_setCB\_HallSensor(p\_fHallSensorCallback\_t p\_fHallSensorCallback\_t\_in): Sets the Hall Sensor interrupt callback function.

**DRV (Hardware Driver Layer)**

void drv\_ledInit(): Configures the LED pin as an output.

void drv\_hallInit(): Configures the Hall Sensor pin as an input with an interrupt.

void drv\_blinkLED(): Blinks the LED.

void Port\_1\_ISR(): Interrupt service routine for the Hall Sensor pin. Calls the HAL callback function and increments the pulse count.

**Main**

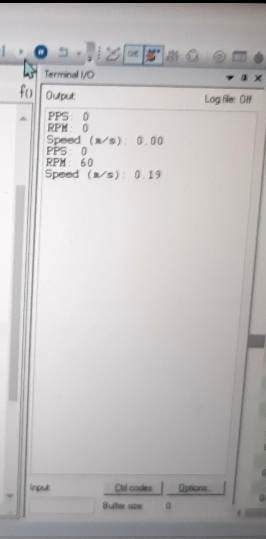
The main application initializes the LED and Hall Sensor, sets the wheel diameter, and enables interrupts. It contains an infinite loop to periodically calculate RPM, PPS, and speed using the functions provided by the HAL. The application also handles the Hall Sensor interrupt to blink the LED and count the pulses.

# Conclusion

The modularized code provides an organized and maintainable library for measuring RPM, Pulse per second, and Speed in m/s using a Hall Sensor. The HAL layer abstracts the hardware specifics, while the DRV layer directly interacts with the hardware. The main application leverages these layers to perform the desired functionalities. The modular approach makes it easier to understand and extend the code, promoting code reusability and readability.

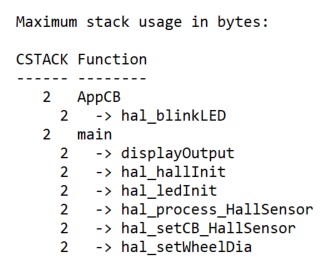
# Results

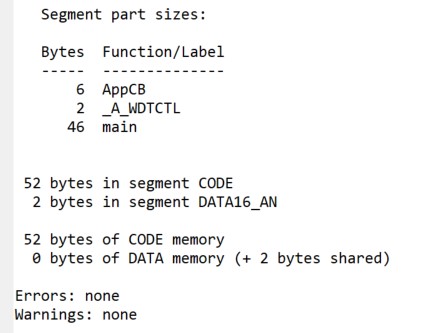
**Output:**





**.lst file**:





**.map file:**

