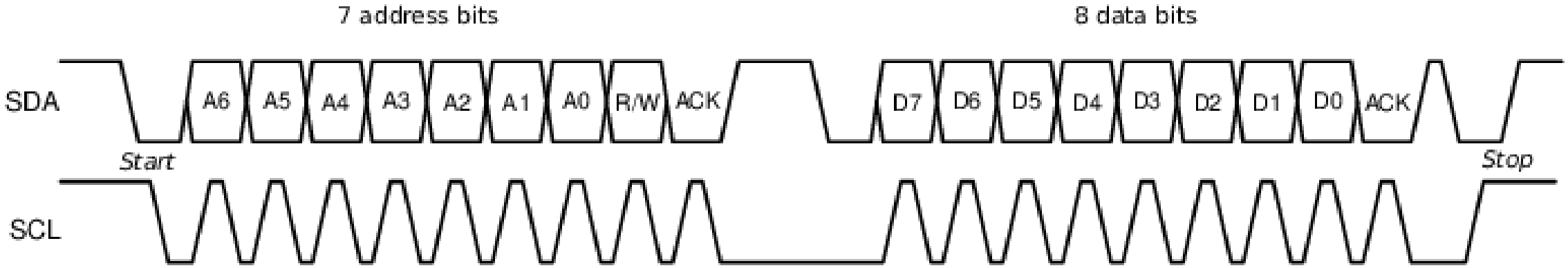
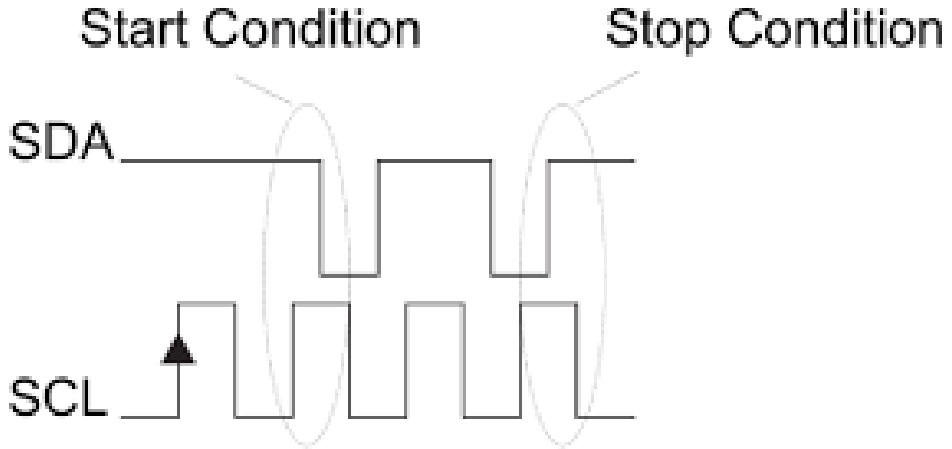
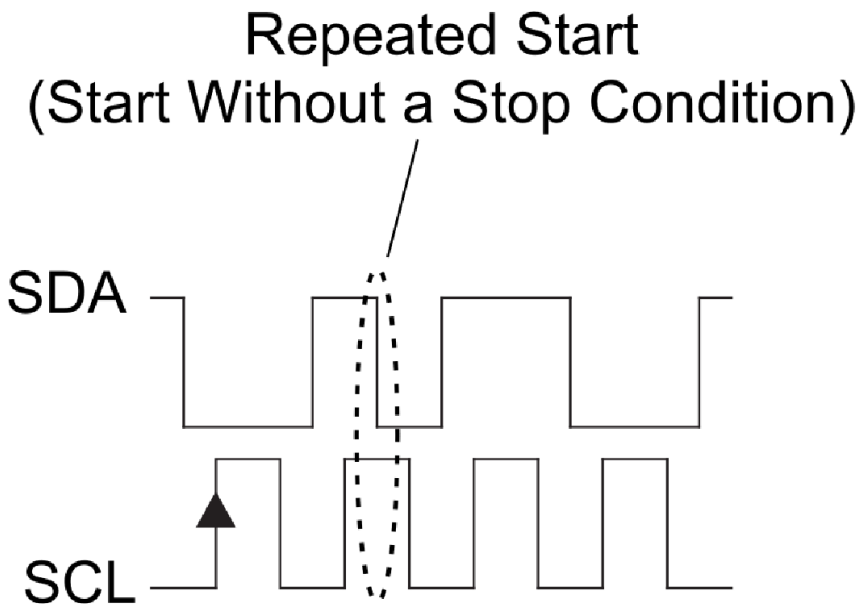
Module I2C
Capstone Project

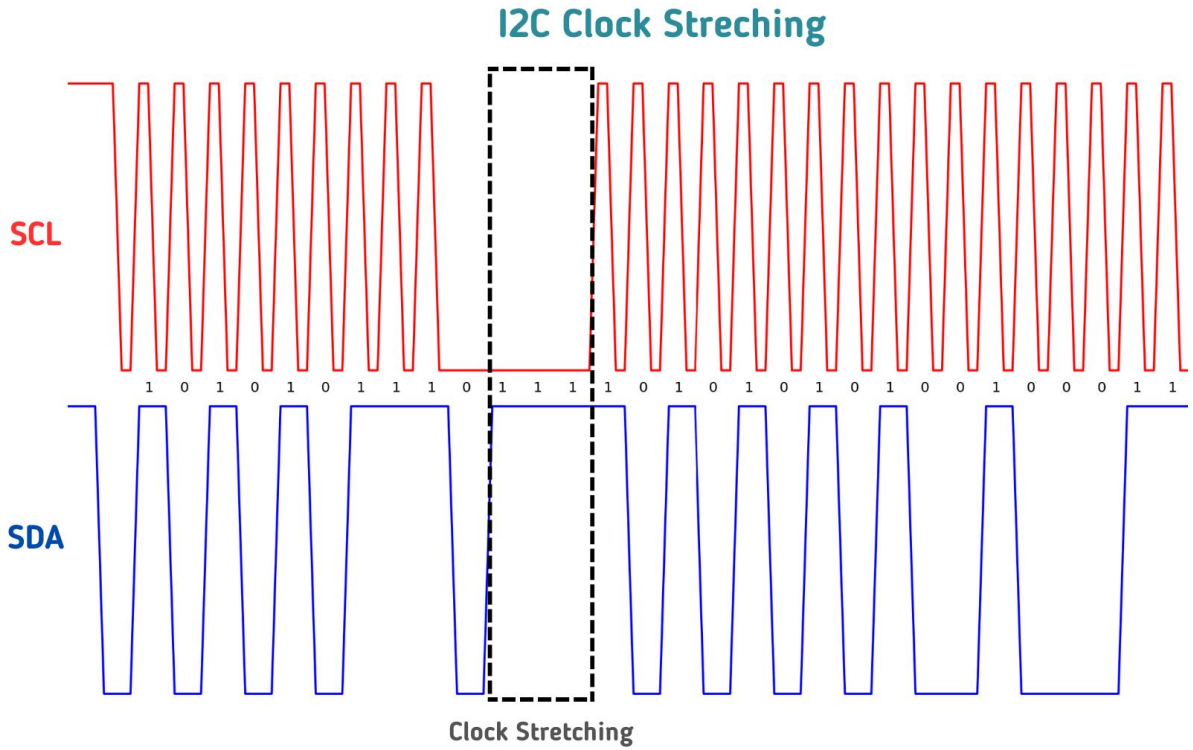
I2C Timing Diagram





I2C Timing Diagram





Register Sequence

- Configure the Clock for – GPIO, I2C
- Configure the GPIO for ALT functionality with open drain and pull up configuration
- Software reset the I2C peripheral - Set and then clear the SWRST bit in I2C_CR1 register
- Configure I2C clock frequency - Set FREQ bits in I2C_CR2 to match APB1 clock frequency in MHz
- Configure I2C clock control register (CCR) For standard mode (100 kHz):
  - Clear FS bit in I2C_CCR
  - Set CCR value = (APB1_FREQ / (2 * I2C_FREQ))
- Configure maximum rise time For standard mode:
  - TRISE = (maximum_rise_time_in_ns / APB1_period_in_ns) + 1
- Set own address (if configuring for slave)
  - Clear ADDMODE bit in I2C_OAR1 for 7-bit mode
  - Set ADD[7:1] bits with the address
  - Set bit 14 in I2C_OAR1 (must be kept at 1 by software)
- Enable I2C peripheral
  - Set PE bit in I2C_CR1 register

Transmitter Sequence

Thynk Loop

- Wait for the bus to be free (not busy)
- Generate START condition by setting the START bit
- Wait until START condition is successfully generated (check SB flag)
-
- Send slave address with Write bit (LSB = 0)
- Wait until address is sent and ACK received (check ADDR flag)
- Clear ADDR flag by reading SR1 and SR2 registers
- Send the data byte by writing to DR register
- Wait until the data byte has been transmitted (check TXE flag)
- If more bytes to send, go to step 7
- Wait until the last byte transfer is complete (check BTF flag)
- Generate STOP condition by setting the STOP bit

Capstone Project – Smart Automation

Thynk Loop

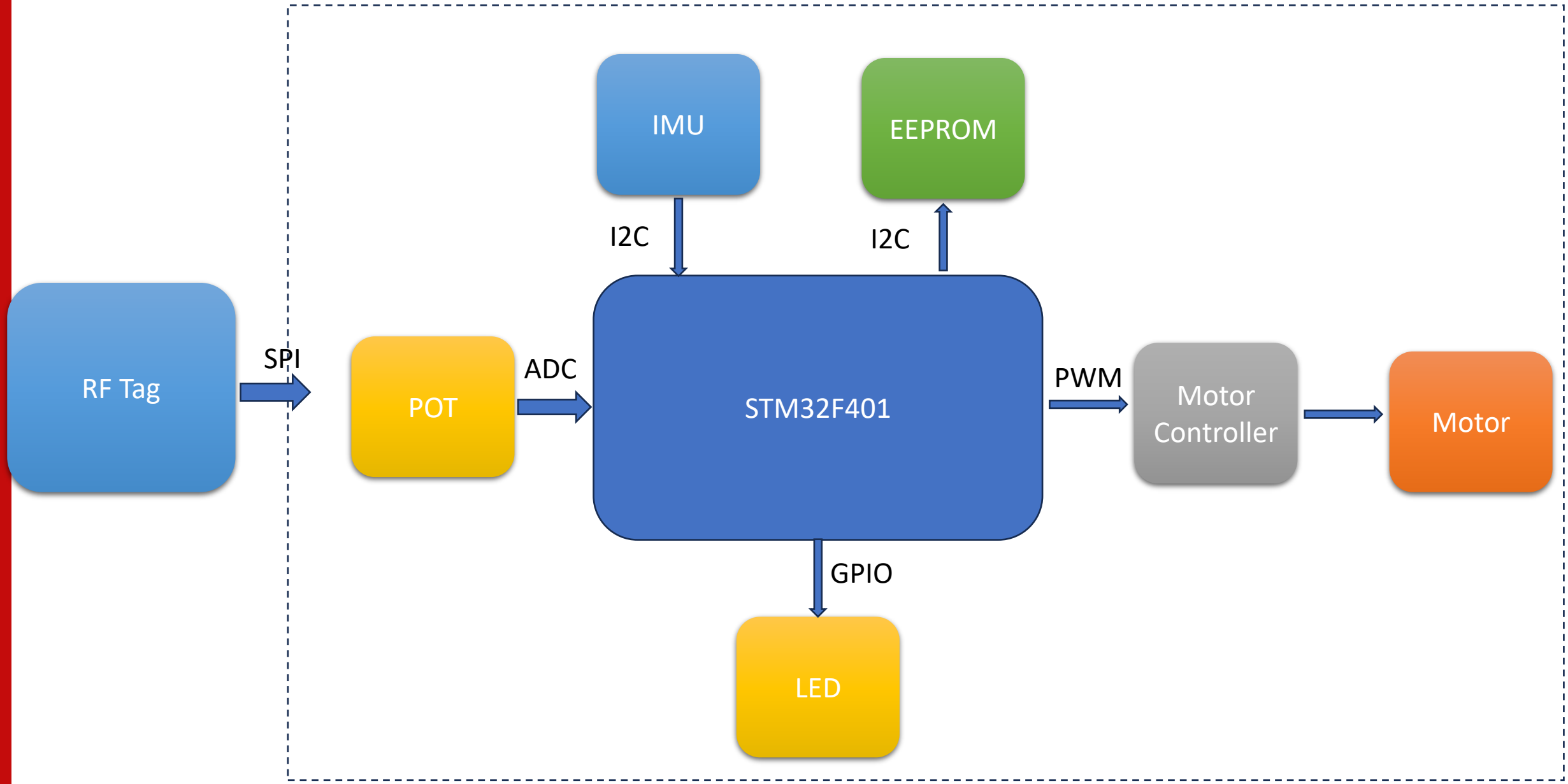
Project Overview

•The system is designed to enable and control peripherals only when a valid RF tag is detected.
•It uses an STM32F401 microcontroller as the central processing unit.
•Communication interfaces include SPI for RF tag authentication, I2C for sensor data acquisition and storage, ADC for speed control, and PWM for motor control.
•The system deactivates all peripherals when the RF tag is removed.

Hardware Components

•**Microcontroller:** STM32F401 (acts as the main controller).
•**RF Tag & Reader:** Used for authentication (connected via SPI).
•**Potentiometer (POT):** Used to control motor speed (connected to ADC).
•**IMU Sensor:** Captures motion data (connected via I2C).
•**EEPROM:** Stores IMU data (connected via I2C).
•**Motor & Motor Controller:** Controls motor speed and direction using PWM.
•**LED Indicator:** Turns on when authentication is successful (controlled via GPIO).

Capstone Project – Smart Automation



Capstone Project – Smart Automation

Thynk Loop

Functional Requirements

RF Tag Authentication:

- The RF reader communicates with STM32F401 via SPI.
- Only a valid RF tag enables the entire system.
- If authentication fails, all peripherals remain inactive.

LED Control (GPIO):

- If authentication is successful, the LED turns ON.
- If the RF tag is removed, the LED turns OFF.

Motor Speed Control (ADC & PWM):

- A potentiometer provides an analog voltage input to the ADC.
- The microcontroller converts this value into a PWM signal.
- The motor controller receives the PWM signal and adjusts the motor speed accordingly.

IMU Data Acquisition (I2C):

- The IMU sensor collects motion data.
- Data is periodically read via the I2C interface.

Data Logging in EEPROM (I2C):

- IMU data is stored in an EEPROM connected via I2C.
- The system ensures proper data logging while authentication is active.

System Deactivation:

- If the RF tag is removed, all peripherals (LED, motor, IMU logging) are immediately disabled.

Capstone Project – Smart Automation

- Implement all the given functional requirements
- Once implemented, PUSH the code in Github. Kindly attach the short video of the same as well in Github (or link to the video)
- Feel free to approach us for any questions