ieee**cusb**

Embedded AVR Workshop

Final Project

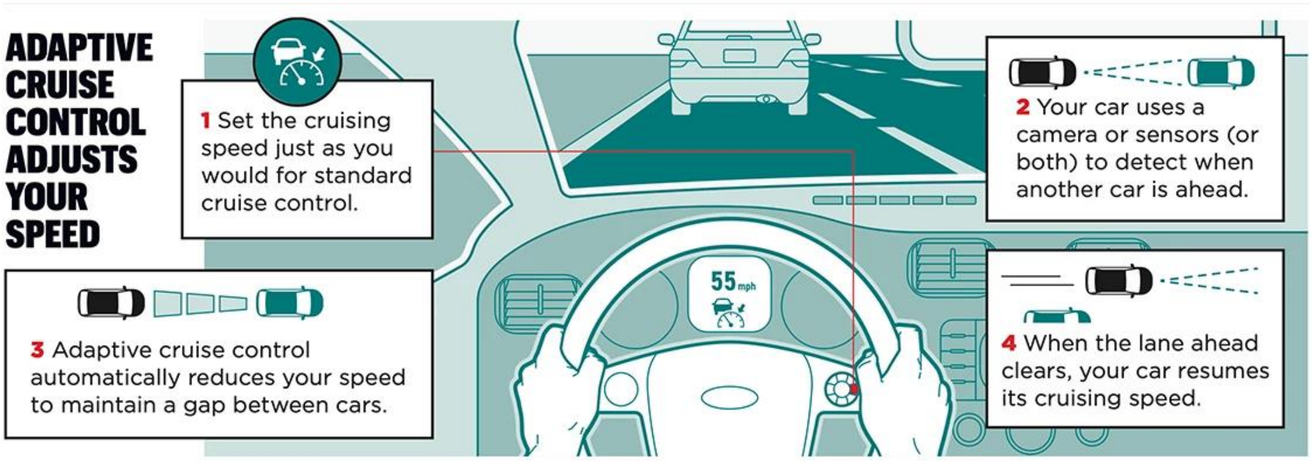
2024-2025

IEEE CAIRO UNIVERSITY STUDENT BRANCH

Adaptive Cruise Control

Automotive Application

The **Adaptive Cruise Control (ACC) Simulator** is a microcontroller-based project designed to emulate a key Advanced Driver Assistance System (ADAS) feature: maintaining a safe following distance while preserving a user-defined cruising speed. The system is implemented on an **ATmega32 microcontroller** and integrates real-time sensor data, actuator control, and user interaction to replicate the logic of an automotive ACC system in a controlled, educational environment.



System Requirements

1. Use ATmega32 Microcontroller with frequency 16Mhz.

2. The project should be design and implemented based on the layered architecture model as follow:

Application Layer

HAL

(Buzzer – DC Motor – LCD – Push Button – Ultrasonic Sensor)

MCAL

(ADC – GPIO – ICU – PWM)

Application Layer

Functional Requirements

- **Speed Input**: Read an analog input via ADC to represent the desired speed (0–100 km/h) using a potentiometer

- **Distance Measurement**: Use the ultrasonic sensor to measure the distance to an obstacle in centimeters.

- **ACC Logic**:
  - <u>When ACC is enabled</u>:
    - ✓ Stop the motor and activate the buzzer if the obstacle is closer than 20 cm.
    - ✓ Set motor speed to 20% of input speed if the distance is 20–50 cm.
    - ✓ Set motor speed to 50% of input speed if the distance is 50–100 cm.
    - ✓ Use full input speed if the distance is ≥100 cm.

  - <u>When ACC is disabled</u>: Set motor speed directly to the input speed, with no buzzer.

- **Toggle ACC**: Use a push button (via INT0 interrupt) to enable/disable ACC mode. It is initially enabled.

- **Display**: Show real-time speed (km/h) and distance (cm) on the LCD.

Buzzer Driver

HAL Layer

- `void Buzzer_init()`
  - It initializes the buzzer pin as an output pin, and turns it off at the beginning.

  - Connect the buzzer to pin PB5.

- `void Buzzer_ON()`
  - It turns the buzzer on.

- `void Buzzer_OFF()`
  - It turns the buzzer off.

DC Motor

HAL Layer

- **void DcMotor_Init(void)**
  - ➢ The Function responsible for setup the direction for the two motor pins through the GPIO driver.
  - ➢ Stop at the DC-Motor at the beginning through the GPIO driver.
  - ➢ Connect IN1 & IN2 to PD0 & PD1 respectively. And the EN pin to PB3.

- **void DcMotor_Rotate(DcMotor_State state,uint8 speed)**
  - ➢ The function responsible for rotate the DC Motor CW/ or A-CW or stop the motor based on the state input state value.
  - ➢ Send the required duty cycle to the PWM driver based on the required speed value.
  - ➢ **Inputs:**
    - ✓ state: -The required DC Motor state, it should be CW or A-CW or stop.
      -DcMotor_State data type should be declared as **enum** or **uint8**.
    - ✓ speed: decimal value for the required motor speed, it should be from 0 → 100. For example, if the input is 50, The motor should rotate with 50% of its maximum speed.

LCD Driver

HAL layer

- Use the exact same driver implemented in the workshop.

- Connect the data pins to PORTC, the enable pin to PB6, and the RS pin to PB7.

Push Button

HAL Layer

- **void BUTTON_init**()
    - It initializes the push button pin as an input pin, and it activates the internal pull-up resistor.
    - Connect the button to pin PD2.

- **uint8 BUTTON_read**()
    - It gets the value, either high or low, of the push button pin and returns it.

Ultrasonic Sensor

HAL Layer

- `void Ultrasonic_init(void)`
  - Initialize the ICU driver as required.
  - Setup the ICU call back function.
  - Setup the direction for the trigger pin as output pin through the GPIO driver, connected to pin PD7.

- `void Ultrasonic_Trigger(void)`
  - Send the Trigger pulse to the ultrasonic.

- `uint16 Ultrasonic_readDistance(void)\`
  - Send the trigger pulse by using Ultrasonic_Trigger function.
  - Start the measurements by the ICU from this moment.
  - Returns the measured distance in Centimeter.

- `void Ultrasonic_edgeProcessing(void)`
  - This is the call back function called by the ICU driver.
  - This is used to calculate the high time (pulse time) generated by the ultrasonic sensor.

MCAL Layer

Microcontroller Peripherals

- **ADC Driver**
  - Use the exact same driver implemented in the workshop.
  - Connect the potentiometer to ADC0.
- **GPIO Driver**
  - Use the exact same driver implemented in the workshop.
- **ICU Driver**
  - Use the exact same driver implemented in the workshop.
  - The ICU should be configured with frequency F_CPU/8 and to detect the raising edge as the first edge.
  - ICU_init and ICU_setCallBack functions should be called inside the Ultrasonic_init function.

PWM Driver

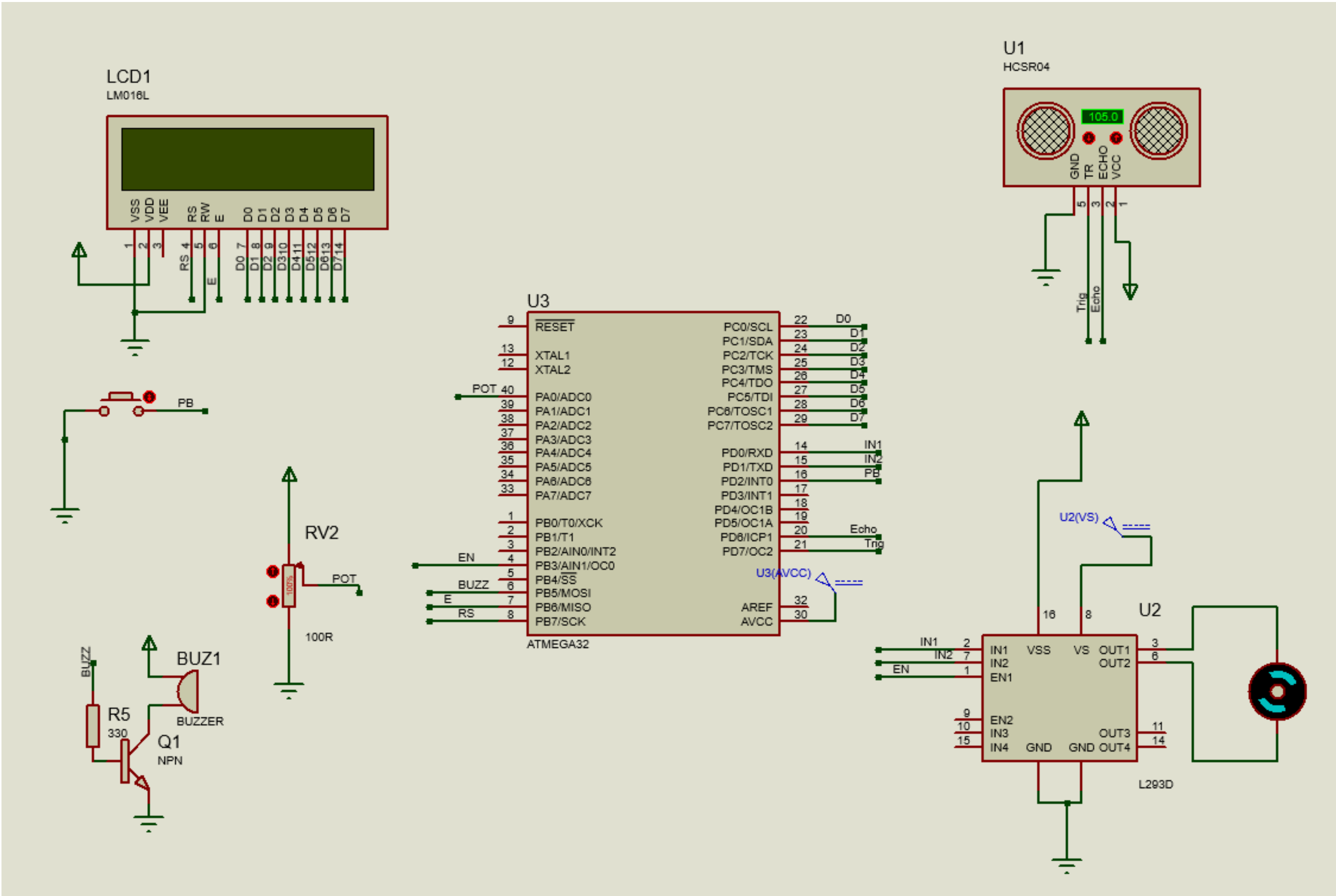
MCAL Layer

- The one implemented in the workshop was implemented using Timer1 module, which will be reserved for the ICU functionality. So you will have to implement it again using Timer0.

- `void PWM_Timer0_Start(uint8 duty_cycle)`
  - The function responsible for trigger the Timer0 with the PWM Mode.
  - Setup the PWM mode with Non-Inverting.
  - Setup the prescaler with F_CPU/8.
  - Setup the compare value based on the required input duty cycle.
  - Setup the direction for OC0 as output pin through the GPIO driver.
  - The generated PWM signal frequency will be 7.8125Hz to control the DC Motor speed.

Project Hardware

Proteus Simulation



Project Deliverables

What to submit?

- You should submit only one ZIP file containing the proteus simulation, and the eclipse project, containing all the software drivers and the application code.

- **Deadline of submission:** 18/7/2025

Good Luck!