Capacity Building in Teaching of AR/VR (CATCH_VR)
European Commission Project# 101129191 (ERASMUS-EDU-2022-CBHE)

Co-funded by the European Union

Microcontrollers & Embedded Systems Digital Twin

Conducted by. Dr. Gulbadan Sikandar, 16th November, 2025
Department of Mechatronics Engineering, UET Peshawar

Digital twin conversion pattern

Each legacy embedded lab becomes a closed-loop physical + virtual experiment.

Physical system

Arduino / ATmega328P
Sensors, LEDs, motors, displays

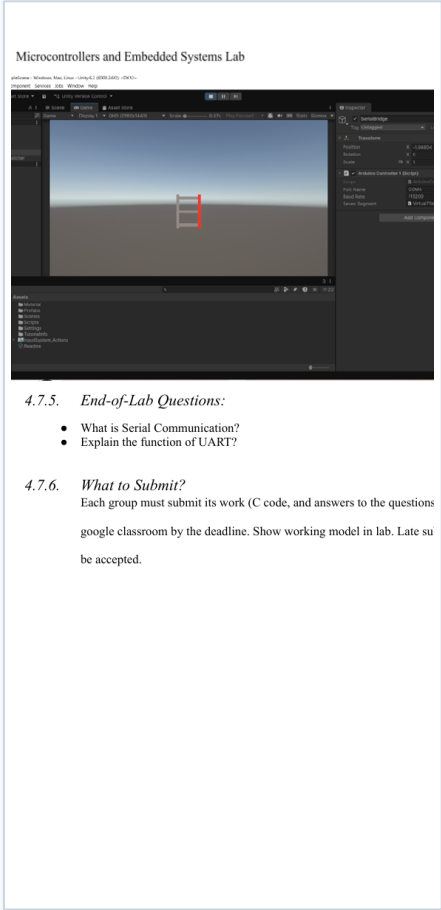
Serial protocol

9600 baud UART
Commands + state telemetry

Unity digital twin

Animated virtual asset
HIL observation and control

Standardize every experiment around: objectives -> hardware circuit -> embedded program -> serial message schema -> Unity DT behavior -> evidence screenshots/video + lab questions.


Microcontrollers and Embedded Systems Lab

4.7.5. *End-of-Lab Questions:*

- What is Serial Communication?
- Explain the function of UART?

4.7.6. *What to Submit?*

Each group must submit its work (C code, and answers to the questions

google classroom by the deadline. Show working model in lab. Late su

be accepted.

Unity serial-linked seven-segment DT shown in the manual, page 50.

C Programming in Microchip Studio

Build, compile, and generate HEX files for ATmega programs.

Foundation lab: toolchain confidence before hardware-in-the-loop work.

CORE OBJECTIVE

- Navigate Microchip Studio IDE
- Write and compile AVR C source code
- Generate HEX files for flashing

PHYSICAL / SIMULATION BUILD

- Install/check Microchip Studio
- Create a project and enter LED blink code
- Modify delay to observe blink-rate effect

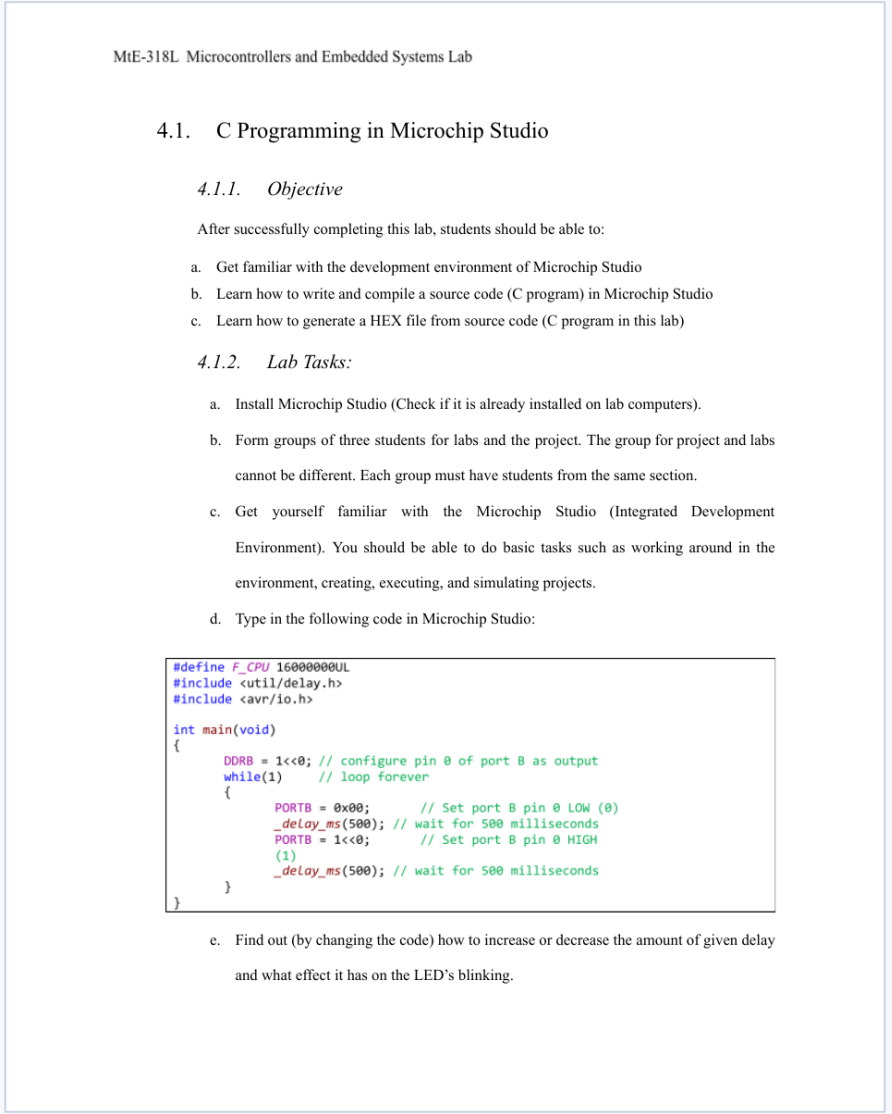
DIGITAL TWIN CONVERSION

- No Unity asset required yet
- Produces firmware used later in Proteus/Arduino DT labs
- Introduces repeatable compile-to-flash workflow

ASSESSMENT EVIDENCE

- C source code
- Delay-change observations
- Answers on burning code and DDR/PORT commands

Digital twin protocol: Toolchain output: .hex firmware for later simulated/physical twins



MtE-318L Microcontrollers and Embedded Systems Lab

4.1. C Programming in Microchip Studio

4.1.1. Objective

After successfully completing this lab, students should be able to:

a. Get familiar with the development environment of Microchip Studio
b. Learn how to write and compile a source code (C program) in Microchip Studio
c. Learn how to generate a HEX file from source code (C program in this lab)

4.1.2. Lab Tasks:

a. Install Microchip Studio (Check if it is already installed on lab computers).
b. Form groups of three students for labs and the project. The group for project and labs cannot be different. Each group must have students from the same section.
c. Get yourself familiar with the Microchip Studio (Integrated Development Environment). You should be able to do basic tasks such as working around in the environment, creating, executing, and simulating projects.
d. Type in the following code in Microchip Studio:

```
#define F_CPU 16000000UL
#include <util/delay.h>
#include <avr/io.h>

int main(void)
{
	DDRB = 1<<0; // configure pin 0 of port B as output
	while(1)     // loop forever
	{
		PORTB = 0x00;       // Set port B pin 0 LOW (0)
		_delay_ms(500); // wait for 500 milliseconds
		PORTB = 1<<0;       // Set port B pin 0 HIGH
		(1)
		_delay_ms(500); // wait for 500 milliseconds
	}
}
```

e. Find out (by changing the code) how to increase or decrease the amount of given delay and what effect it has on the LED's blinking.

LED blink source-code task, manual page 17.

Getting Started with Proteus Professional

Create and simulate microcontroller circuits before deploying hardware.

Simulation lab: validate LED behavior in Proteus before physical or DT coupling.

CORE OBJECTIVE

- Create Proteus projects
- Build ATmega328P LED circuits
- Run circuit simulations

PHYSICAL / SIMULATION BUILD

- Connect LED to PB4 / Arduino pin 12
- Load HEX from Lab 1
- Extend to four-LED sequences

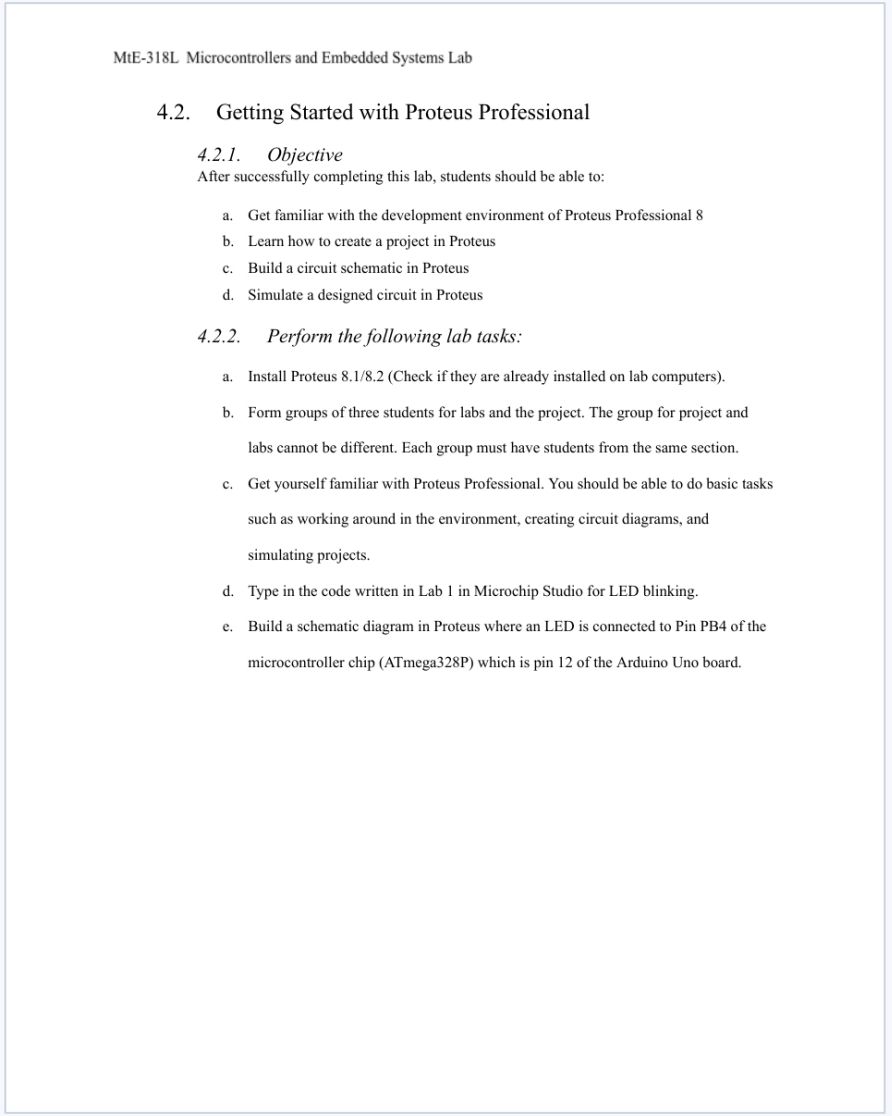
DIGITAL TWIN CONVERSION

- Proteus acts as pre-DT virtual validation
- Confirms firmware/circuit behavior before Unity linkage
- Separates logic errors from DT communication errors

ASSESSMENT EVIDENCE

- Proteus circuit file
- Simulation screenshots
- Answers on basic Proteus workflow

Digital twin protocol: Virtual validation: firmware + schematic simulation



MtE-318L Microcontrollers and Embedded Systems Lab

4.2. Getting Started with Proteus Professional

4.2.1. Objective

After successfully completing this lab, students should be able to:

a. Get familiar with the development environment of Proteus Professional 8
b. Learn how to create a project in Proteus
c. Build a circuit schematic in Proteus
d. Simulate a designed circuit in Proteus

4.2.2. Perform the following lab tasks:

a. Install Proteus 8.1/8.2 (Check if they are already installed on lab computers).
b. Form groups of three students for labs and the project. The group for project and labs cannot be different. Each group must have students from the same section.
c. Get yourself familiar with Proteus Professional. You should be able to do basic tasks such as working around in the environment, creating circuit diagrams, and simulating projects.
d. Type in the code written in Lab 1 in Microchip Studio for LED blinking.
e. Build a schematic diagram in Proteus where an LED is connected to Pin PB4 of the microcontroller chip (ATmega328P) which is pin 12 of the Arduino Uno board.

Proteus objectives and tasks, manual page 20.

ATmega328P Pin-out and Datasheet

Map pins, use Arduino IDE, and link physical LED state to a DT LED.

Students move from pin mapping to serially controlled Unity LED visualization.

CORE OBJECTIVE

- Understand ATmega328P pin mapping
- Implement LED blinking and fading
- Use analogRead, analogWrite, and map()

PHYSICAL / SIMULATION BUILD

- Arduino UNO, breadboard, LED, potentiometer
- Blink LED, fade LED, control brightness via pot
- Compare pin names to Arduino board pins

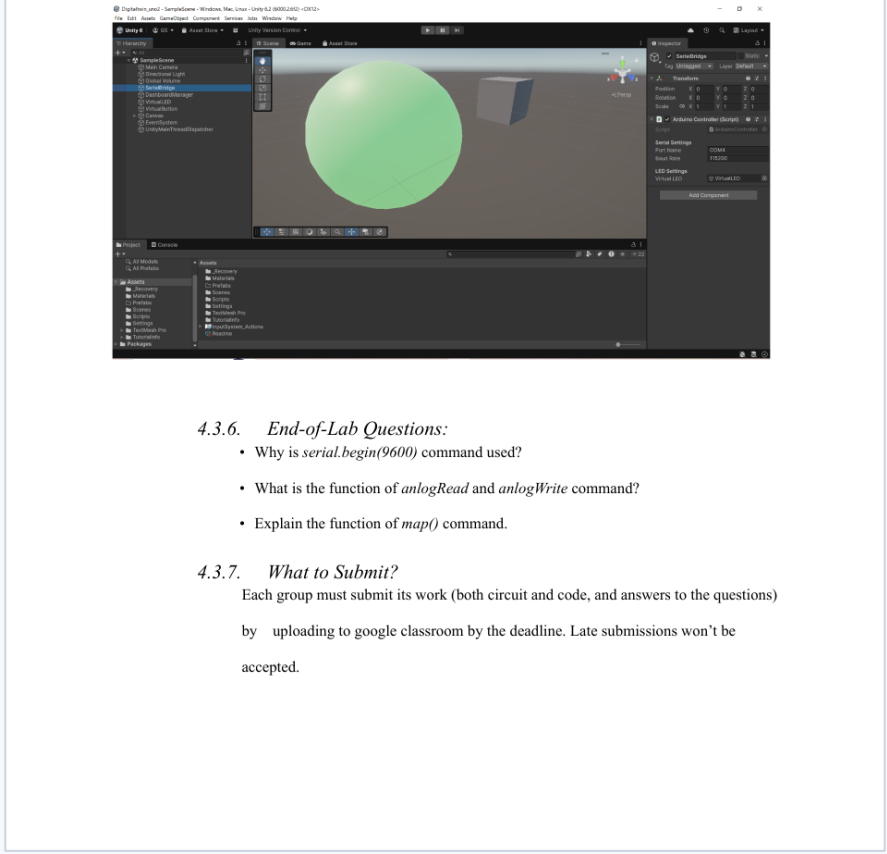
DIGITAL TWIN CONVERSION

- Open Unity DTLED project
- Real Arduino sends state to DT LED
- DT LED mirrors physical LED ON/OFF

ASSESSMENT EVIDENCE

- Circuit, Arduino code, screenshots
- Answers on Serial.begin, analogRead/write, map
- Demonstrated physical + DT LED behavior

Digital twin protocol: LED_ON / LED_OFF at 9600 baud



MtE-318L Microcontrollers and Embedded Systems Lab

4.3.5. Digital Twin based Experiment

Perform the LED blinking experiment on the Digital Twin, where the real Arduino serially sends a signal to the DT LED.

- Open Unity Hub and click on the DTLED project
- Make sure the baud rate is set at 9600 in the Arduino code
- Send LED_ON to turn on the DTLED and LED_OFF to turn it off.

4.3.6. End-of-Lab Questions:

- Why is *serial.begin(9600)* command used?
- What is the function of *anlogRead* and *anlogWrite* command?
- Explain the function of *map()* command.

4.3.7. What to Submit?

Each group must submit its work (both circuit and code, and answers to the questions) by uploading to google classroom by the deadline. Late submissions won't be accepted.

Unity DTLED task and scene, manual page 26.

Switch Debouncing

Debounce a pushbutton and use it as a DT control event.

A button press becomes a reliable event for both physical and virtual LEDs.

CORE OBJECTIVE

- Interface a pushbutton with AVR
- Recognize mechanical bounce
- Implement software debouncing

PHYSICAL / SIMULATION BUILD

- Build switch + LED circuit in Proteus and breadboard
- Run non-debounced and debounced programs
- Modify logic so LED toggles on every valid press

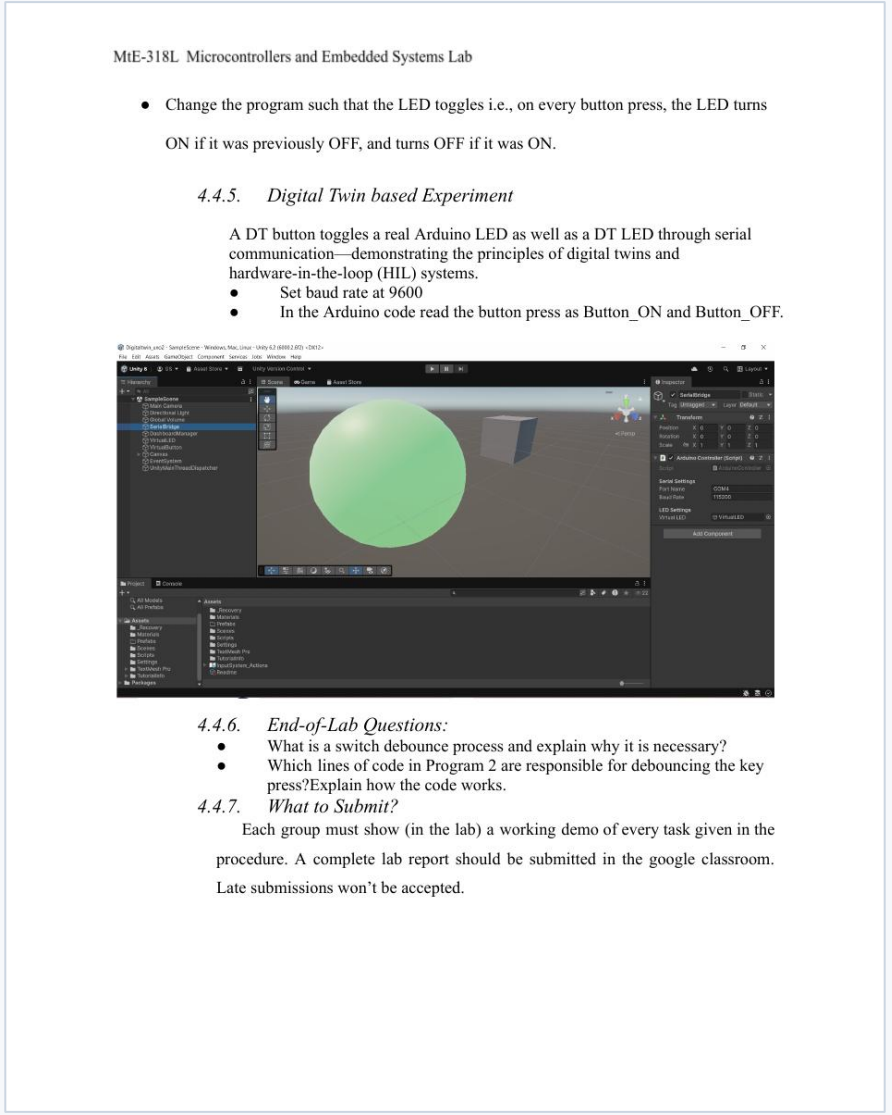
DIGITAL TWIN CONVERSION

- DT button toggles real Arduino LED
- Same event toggles Unity DT LED
- Demonstrates hardware-in-the-loop event flow

ASSESSMENT EVIDENCE

- Working breadboard demo
- Complete lab report
- Explanation of debounce code lines

Digital twin protocol: Button_ON / Button_OFF at 9600 baud



MtE-318L Microcontrollers and Embedded Systems Lab

- Change the program such that the LED toggles i.e., on every button press, the LED turns ON if it was previously OFF, and turns OFF if it was ON.

4.4.5. Digital Twin based Experiment

A DT button toggles a real Arduino LED as well as a DT LED through serial communication—demonstrating the principles of digital twins and hardware-in-the-loop (HIL) systems.

- Set baud rate at 9600
- In the Arduino code read the button press as Button_ON and Button_OFF.

4.4.6. End-of-Lab Questions:

- What is a switch debounce process and explain why it is necessary?
- Which lines of code in Program 2 are responsible for debouncing the key press?Explain how the code works.

4.4.7. What to Submit?

Each group must show (in the lab) a working demo of every task given in the procedure. A complete lab report should be submitted in the google classroom. Late submissions won't be accepted.

Button-to-DT LED task, manual page 31.

4.5. Timers in AVR Microcontroller

Use Timer0 normal mode for accurate delay generation and DT status updates.

Hardware timers replace blocking delay loops and stabilize observable twin behavior.

CORE OBJECTIVE

- Configure Timer0 in normal mode
- Use prescaler-based timing
- Toggle LED with timer-generated delay

PHYSICAL / SIMULATION BUILD

- ATmega328P, LED, resistor, Proteus
- Prescaler = 64, Timer0 overflow timing
- Generate about 500 ms LED toggle

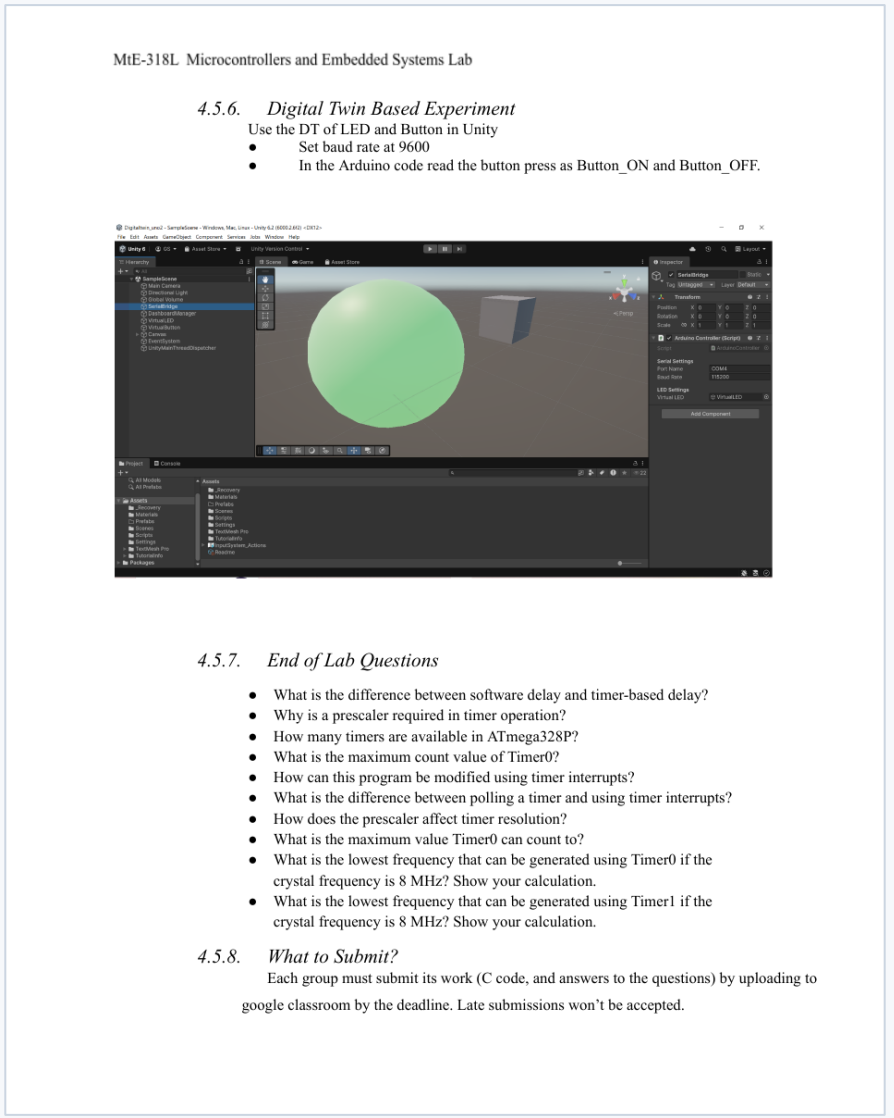
DIGITAL TWIN CONVERSION

- Use Unity LED/Button DT scene
- Button states exchanged over serial
- Timer-driven outputs keep DT timing consistent

ASSESSMENT EVIDENCE

- C code and timer calculations
- Answers on prescalers, timer count, interrupts
- Simulation/hardware evidence

Digital twin protocol: Button_ON / Button_OFF at 9600 baud



MtE-318L Microcontrollers and Embedded Systems Lab

4.5.6. Digital Twin Based Experiment

Use the DT of LED and Button in Unity

- Set baud rate at 9600
- In the Arduino code read the button press as Button_ON and Button_OFF.

4.5.7. End of Lab Questions

- What is the difference between software delay and timer-based delay?
- Why is a prescaler required in timer operation?
- How many timers are available in ATmega328P?
- What is the maximum count value of Timer0?
- How can this program be modified using timer interrupts?
- What is the difference between polling a timer and using timer interrupts?
- How does the prescaler affect timer resolution?
- What is the maximum value Timer0 can count to?
- What is the lowest frequency that can be generated using Timer0 if the crystal frequency is 8 MHz? Show your calculation.
- What is the lowest frequency that can be generated using Timer1 if the crystal frequency is 8 MHz? Show your calculation.

4.5.8. What to Submit?

Each group must submit its work (C code, and answers to the questions) by uploading to google classroom by the deadline. Late submissions won't be accepted.

Timer lab DT extension, manual page 35.

DC Motor Control

Control motor speed/direction with PWM and motor-driver hardware.

A motor-control lab built around safe H-bridge/L298 operation and PWM duty cycle.

CORE OBJECTIVE

- Describe DC motor operation
- Generate PWM for speed control
- Drive direction through H-bridge logic

PHYSICAL / SIMULATION BUILD

- Implement transistor H-bridge and L298/L298D circuit
- Control forward/reverse/stop states
- Analyze duty cycle and PWM frequency

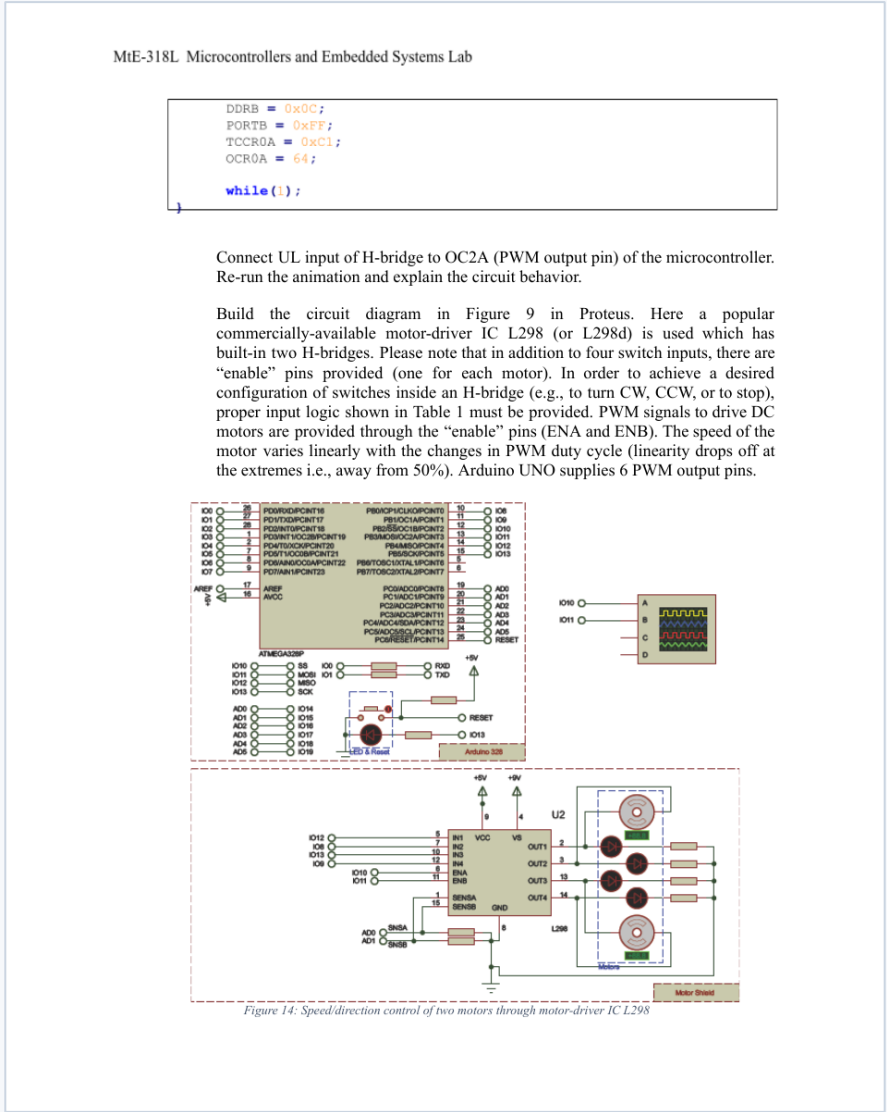
DIGITAL TWIN CONVERSION

- Create/mirror DT motor state in Unity
- Send speed, direction, and stop commands
- Use HIL to visualize unsafe vs safe driver states

ASSESSMENT EVIDENCE

- Commented C code for Programs 1 and 2
- PWM calculations for TCNT/TCCR/OCR registers
- Working motor driver demo

Digital twin protocol: Recommended: MOTOR:DIR,SPEED and STATE telemetry



L298 dual-motor control schematic, manual page 42.

Serial Communication

Use UART as the backbone of physical-to-digital-twin synchronization.

Students establish the serial link used by all later DT experiments.

CORE OBJECTIVE

- Configure UART communication
- Send and receive serial data
- Link microcontroller, PC, and terminal software

PHYSICAL / SIMULATION BUILD

- Use Tera Term and UART-to-TTL/USB interface
- Transmit OK messages
- Receive PC commands to drive LEDs / seven-segment

DIGITAL TWIN CONVERSION

- Unity seven-segment display receives digit messages
- Serial strings become DT display state
- Forms the canonical HIL messaging pattern

ASSESSMENT EVIDENCE

- C code and screenshots
- 7-segment UART interrupt task
- Answers on serial communication and UART

Digital twin protocol: DIGIT:<value> at 9600 baud

MtE-318L Microcontrollers and Embedded Systems Lab

```c
#define F_CPU 1000000
#include <avr/io.h>
#include <util/delay.h>
//function to initialize UART communication
void UART_initializer()
{
  UBRRL = 0xC;  // set Baud Rate to 4800
  UCSRB |= (1<<RXEN); // enable receiver
  UCSRC |= (1<<URSEL)|(1<<UCSZ0)|(1<<UCSZ1); // set data size
}
//function to receive data over UART
void UART_receive()
{
  while( !( UCSRA & (1<<UDRE) ) ); //poll UDRE and wait to receive data
  char val = UDR;
  if(val >= '0' && val < '8') //check the ASCII sent and
      PORTB = 1 << (val - '0'); //turn on the respective LED
}
int main(void)
{
  DDRB = 0XFF;   //initial configuration for PORTB
  PORTB = 0xFF;
  UART_initializer(); //initialize UART
  while(1)
  {
      UART_receive(); //receive data
  }
}
```

Tasks

Task A: Establish connection between your ATmega328 and PC through UART-to-TTL converter. Run the programs above and explain your observations through screenshots.

Task B: Connect a 7-segment-display with your ATmega328. Establish connection between your ATmega328 and PC through UART-to-TTL converter. Write a program in C such that:

i. Initially, ZERO should be displayed on the 7-segment and the program should wait for the next instruction from the PC.

ii. If a 'u' is sent from PC to ATmega16, the next digit on seven segments should be displayed in ascending order. Ascending order (0,1,2,3,...,8,9,0,1,...

iii. If a 'd' is sent from PC to ATmega16, the next digit on seven segments should be displayed in descending order. Descending order (9,8,7,...,2,1,0,9,...)

iv. If any character other than 'u' or 'd' is sent, do nothing with 7-segment-display and send \ERR" back to PC in order to notify error. Implement the above functionality using AVR serial interrupts.

4.7.4. Digital Twin Based Experiment

Use the DT of Seven Segment Display in Unity

- Set baud rate at 9600
- In the Arduino code send a message as "DIGIT:".

Serial DT seven-segment message format, manual page 49.

LCD Interfacing

Display microcontroller data on both a physical LCD and Unity DT display.

The LCD lab becomes a text-output channel for DT dashboards and status messages.

CORE OBJECTIVE

- Interface a 16x2 LCD to AVR
- Use 4-bit LCD communication
- Program LCD commands/data in C

PHYSICAL / SIMULATION BUILD

- Connect LCM1602C in 4-bit mode
- Display student ID and modify cursor behavior
- Switch displayed ID using pushbutton

DIGITAL TWIN CONVERSION

- Open LCD-KEYPAD Unity digital twin
- Serially send display messages
- DT LCD mirrors selected physical/user text

ASSESSMENT EVIDENCE

- Lab report and working demo
- Answers on 4-bit vs 8-bit mode, RW, E pin
- Evidence of LCD/DT display output

Digital twin protocol: KEY:x at 9600 baud

MtE-318L Microcontrollers and Embedded Systems Lab

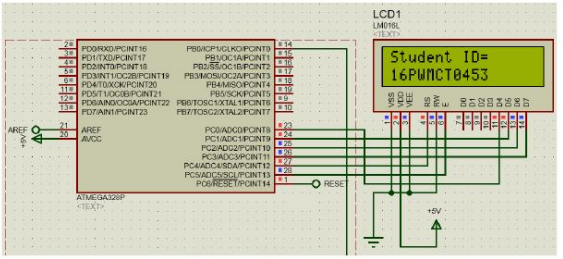
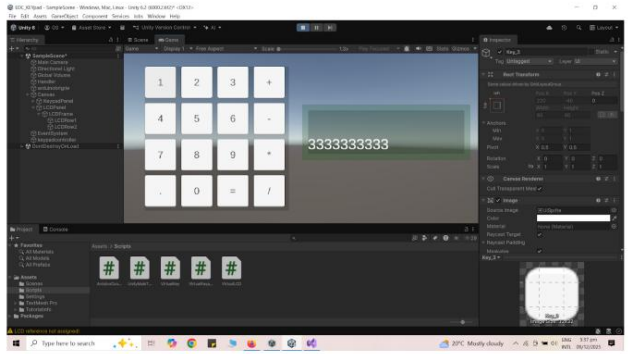


Figure 17

Show a working demon of the circuit task on a breadboard.
Task 1: Change the code to display a blinking cursor at the end of your Student ID. Currently it is off.
Task 2: Change the given program (Program 2) such that every time when a push-button (connected to Port B pin 5) is pressed, the LCD screen displays the Student ID of a different student from your group.

4.8.5. Digital Twin Based Experiment

- Open The LCD-KEYPAD Digital twin in the Unity environment.
- Set baud rate at 9600.
- The message to be displayed should be serial sent as "KEY:x"



LCD DT message requirement, manual page 56.

Keypad Interfacing

Scan matrix keypad inputs and forward key events to Unity.

A keypad becomes the human-input front-end for embedded and DT systems.

CORE OBJECTIVE

- Interface a 4x4 keypad
- Scan rows and columns
- Debounce and transmit key presses

PHYSICAL / SIMULATION BUILD

- Configure rows as outputs and columns as inputs
- Scan one row at a time
- Send pressed characters through USART

DIGITAL TWIN CONVERSION

- Open LCD-KEYPAD Unity digital twin
- Serially send key event as KEY:x
- Use keypad input to drive DT LCD or calculator logic

ASSESSMENT EVIDENCE

- Proteus/breadboard demo
- Separate-line output task and calculator task
- Answers on row scanning and debounce

Digital twin protocol: KEY:x at 9600 baud

MtE-318L Microcontrollers and Embedded Systems Lab

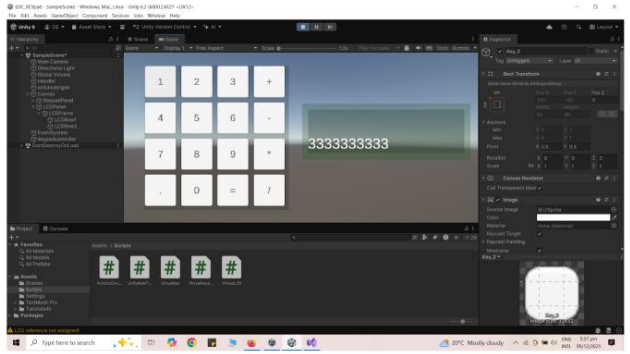
Task 0 (to be done after midterm exam in lab): Show a working demo of the circuit task on a breadboard. The serial window (screen) this time will be on your laptop. The Tera Term software will be used for this purpose.
Task 1: Change the code to display each character on a separate line (as in Figure 5).

Figure 22

Task 2: Implement a basic calculator. For example if someone presses button 7 first, then the "+" button, followed by button 3, you should display 10 (the sum of 3 and 7) in the serial window.

4.9.5. Digital Twin based Experiment

- Open The LCD-KEYPAD Digital twin in the Unity environment.
- Set baud rate at 9600.
- The message to be displayed should be serial sent as "KEY:x"



Keypad-to-DT serial message format, manual page 65.

External Interrupts & PWM LED Control

Handle INT0 events while generating Timer0 PWM fading.

Interrupts provide event-driven twin updates while PWM runs continuously.

CORE OBJECTIVE

- Configure INT0 on PD2
- Write an ISR with software debounce
- Generate PWM on PD6 / OC0A

PHYSICAL / SIMULATION BUILD

- Button on PD2 with internal pull-up
- Toggle LED on PB0 via ISR
- Fade PWM LED using OCR0A updates

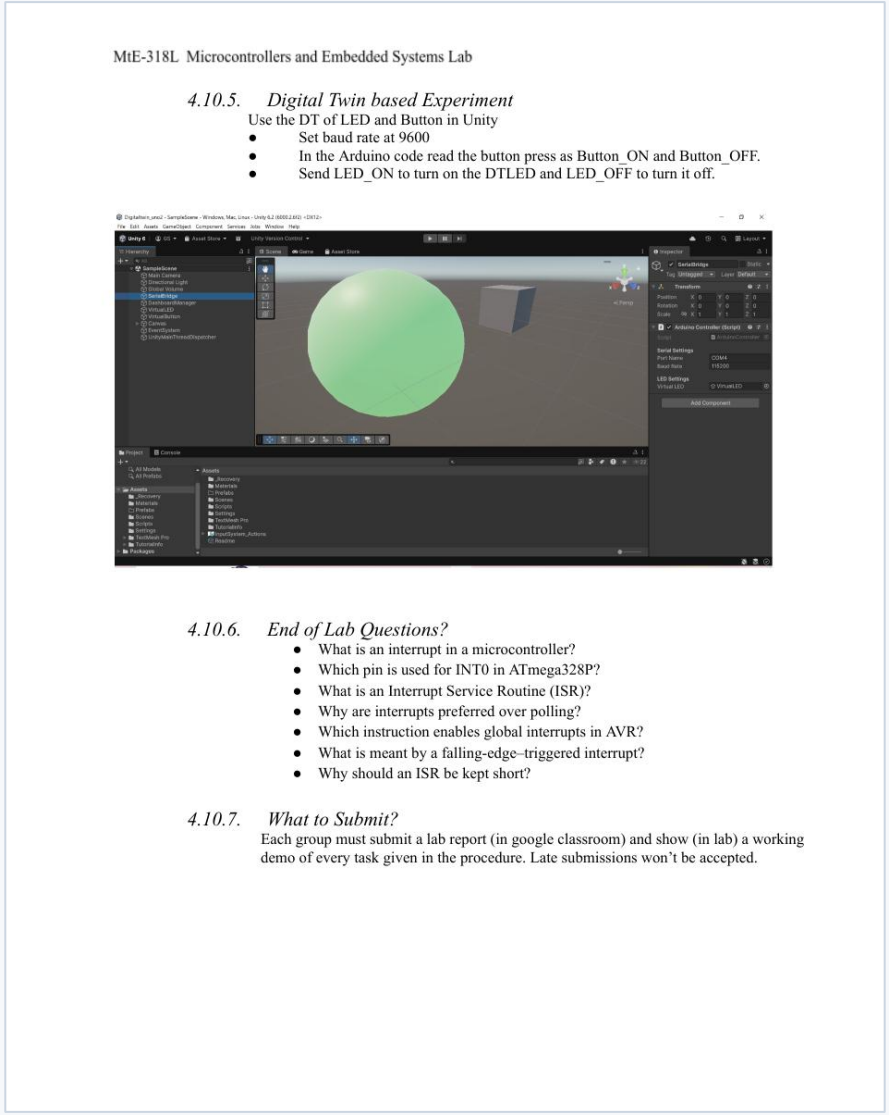
DIGITAL TWIN CONVERSION

- Unity LED/Button scene mirrors interrupt-driven state
- Physical button sends LED_ON/OFF to DT LED
- Shows event-driven HIL response

ASSESSMENT EVIDENCE

- Lab report and Proteus/hardware demo
- Answers on ISR, polling, falling edge, sei()
- DT evidence of button/LED behavior

Digital twin protocol: Button_ON / Button_OFF and LED_ON / LED_OFF at 9600 baud



MtE-318L Microcontrollers and Embedded Systems Lab

4.10.5. Digital Twin based Experiment

Use the DT of LED and Button in Unity

- Set baud rate at 9600
- In the Arduino code read the button press as Button_ON and Button_OFF.
- Send LED_ON to turn on the DTLED and LED_OFF to turn it off.

4.10.6. End of Lab Questions?

- What is an interrupt in a microcontroller?
- Which pin is used for INT0 in ATmega328P?
- What is an Interrupt Service Routine (ISR)?
- Why are interrupts preferred over polling?
- Which instruction enables global interrupts in AVR?
- What is meant by a falling-edge–triggered interrupt?
- Why should an ISR be kept short?

4.10.7. What to Submit?

Each group must submit a lab report (in google classroom) and show (in lab) a working demo of every task given in the procedure. Late submissions won't be accepted.

Interrupt + PWM DT requirements, manual page 70.

4.11. Servo Motors

Generate 50 Hz PWM pulses and command matching real/virtual servo angles.

Servo control is framed as exact angle commands exchanged between Unity and Arduino.

CORE OBJECTIVE

- Interface an RC servo
- Generate servo PWM with Timer1
- Map angle to pulse width

PHYSICAL / SIMULATION BUILD

- Set PB1 / OC1A output
- Use ICR1 as TOP for 20 ms period
- Set OCR1A from 0-180 degrees

DIGITAL TWIN CONVERSION

- Unity slider controls real and DT servo
- Arduino reports actual/new position
- Virtual arm tracks physical command

ASSESSMENT EVIDENCE

- Breadboard demo
- Pulse-width modification task
- Answers on OCR1A, ICR1, power, common ground

Digital twin protocol: SERVO:<angle>, e.g., SERVO:90

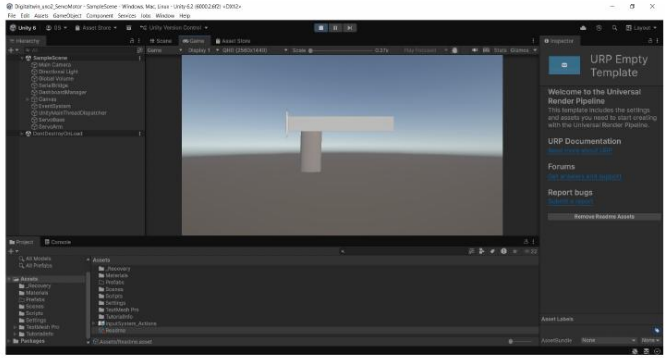
MtE-318L Microcontrollers and Embedded Systems Lab

Task 0 : Show a working demo of the circuit task on a breadboard.

Task 1: Make changes in the pulse width in the code.

4.11.5. Digital Twin based Experiment

- Open the Servo Project in Unity. The slider controls both the real servo arm and the DT version
- This lets Unity send commands like:
  SERVO:90
  SERVO:45
- The servo will move to that exact angle, reporting back its new position.



4.11.6. End-of-Lab Questions:

- Which pin of ATmega328P is used to generate PWM for servo control?
- What is the function of the OCR1A register in servo control?
- What is the role of the ICR1 register in PWM generation?
- Why should the servo motor not be powered directly from the microcontroller?
- Why is a common ground required between the servo and the ATmega328P?
- Why is Fast PWM mode preferred for servo motor control?

4.11.7. What to Submit?

Each group must submit lab report (in google classroom) and show (in la) a working demo of every task given in the procedure. Late submissions won't be accepted.

Servo DT command examples, manual page 73.

Stepper Motor Control

Drive precise step sequences and mirror position in a DT stepper model.

A discrete-motion actuator becomes a position-aware twin through step and position messages.

CORE OBJECTIVE

- Understand stepper motor operation
- Interface motor with ULN2003 driver
- Control speed and direction

PHYSICAL / SIMULATION BUILD

- Connect Arduino D8-D11 to ULN2003 IN1-IN4
- Use full-step sequence
- Reverse order for anticlockwise motion

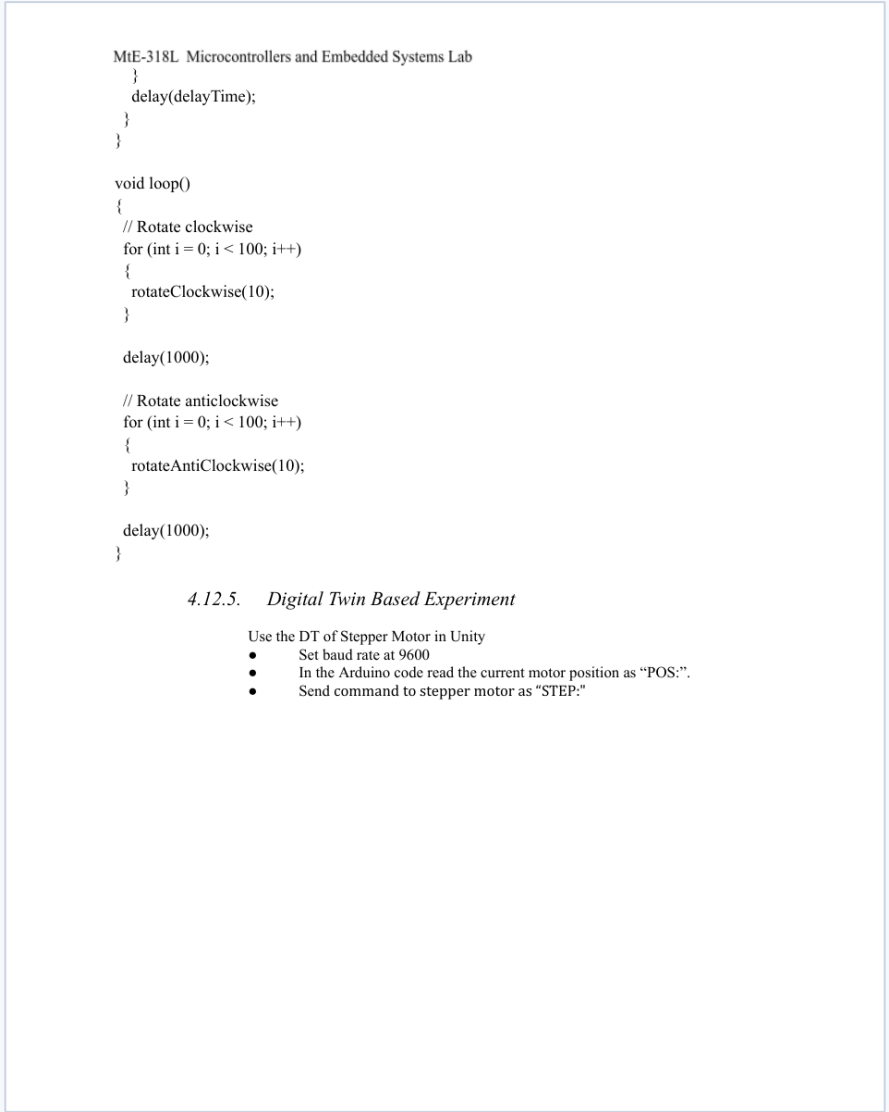
DIGITAL TWIN CONVERSION

- Unity DT receives current motor position
- Arduino sends position telemetry
- Unity or Arduino sends step commands

ASSESSMENT EVIDENCE

- Lab report and working demo
- Answers on driver IC, full/half step, speed and direction
- DT evidence of position tracking

Digital twin protocol: POS:<position> telemetry and STEP:<command>



MtE-318L Microcontrollers and Embedded Systems Lab

```
  }
  delay(delayTime);
 }
}

void loop()
{
 // Rotate clockwise
 for (int i = 0; i < 100; i++)
 {
  rotateClockwise(10);
 }

 delay(1000);

 // Rotate anticlockwise
 for (int i = 0; i < 100; i++)
 {
  rotateAntiClockwise(10);
 }

 delay(1000);
}
```

4.12.5. Digital Twin Based Experiment

Use the DT of Stepper Motor in Unity

- Set baud rate at 9600
- In the Arduino code read the current motor position as "POS:".
- Send command to stepper motor as "STEP:"

Stepper DT message format, manual page 77.