SPC Journal of Education, 5 (1) (2024) 1-22

SPC Journal of Education

Website: www.sciencepubco.com/index.php/JE

Research paperr

Advancing Embedded Systems Education: A Pedagogical Programming Framework for Smart System and Control Applications

Mbadiwe S. Benyeogor[1*], Andrew O. Benyeogor[2*], Kabiru A. Olaiya[3] and Prosper Agumey[*]

[1]*Automata Research Group, Arvin Instruments Nig. Ltd. (RC- 1839472), Ibadan, Nigeria.*
[2]*Computer Science & Robotics Lab., International Leadership of Texas (ILTEXAS), Aggieland, Texas, USA.*
[3]*Department of Mechanical Engineering, Lagos State University of Science and Technology, Lagos, Nigeria.*
[4]*Industrial Liaison Directorate, Accra Technical University, Barnes Road, Accra, Ghana.*
[*]*E-mail: mbadiwebenyeogor@gmail.com[1], abenyeogor1@iltexas.org[2], pagumey@atu.edu.gh[4]*

Abstract

This article introduces a pedagogical framework incorporating a curriculum and conceptual examples, specifically tailored to the need of embedded systems technology instructors The curriculum covers crucial subjects such as hardware and software fundamentals, programming languages, sensors integration, motion control, and their practical applications. It emphasizes hands-on training, project-based learning, and problem-solving to foster a holistic understanding of embedded systems. The curriculum is designed to cater to embedded systems training centers globally, addressing the growing demand for skilled professionals in areas such as robotics, telecommunications, automation, and microprocessor engineering. It aims to bridge the skills gap, empower trainers, and equip students with the necessary competencies to thrive in the evolving field of robotics and programming. Overall, this curriculum serves as a valuable resource for instructors, enabling them to enhance their teaching methods and contribute to the advancement of embedded systems and STEM education.

Keywords: *Curriculum; embedded systems; robotics; programming; STEM education*

1. Introduction

As programming, automation, and robotics evolve rapidly, embedded systems play a crucial role in the development of innovative solutions in these areas (1). To prepare people for skilled employment, it is essential to develop a curriculum or didactic framework that incorporates embedded systems technology in the context of science, technology, engineering, and mathematics (STEM) education (2; 3; 4). Consequently, the present article introduces a curriculum that aims to equip vocational trainers and instructors with the knowledge and skills necessary to effectively teach students concepts in this dynamic and rapidly evolving field of embedded systems, as well as how to apply the acquired skills to areas such as robotics, communication, automation, and microprocessor engineering; thus providing them with a comprehensive framework to deliver high-quality instruction.

By emphasizing hands-on learning and practical applications, vocational trainers can effectively engage students and foster their understanding, technical know-how, and practical application of embedded systems technology. The key topics covered in the curriculum include the fundamentals of embedded systems, programming languages, robotics principles, sensor integration, digital communication, real-time systems, and agricultural automation. Trainers will also learn industry best practices, project-based learning approaches, and assessment strategies to promote an immersive and effective learning experience.

This curriculum is not restricted to a specific geographic area and can be adopted by any training center or institution around the world. By incorporating key concepts and real-world applications, teachers and vocational trainers can empower students to enter the workforce with valuable skills in embedded systems technology, robotics, and programming. Therefore, this curriculum serves as a valuable resource for instructors seeking to improve their teaching methods and provide comprehensive STEM-oriented training in embedded systems technology.

2. Background

By definition, embedded systems are very small computers implanted in consumer appliances to perform control functions that are not noticeable; in other words, they are specialized computing systems embedded within larger devices (5; 6). Examples of such appliances include remote control toy cars, the automobile, the DVD set, the air conditioner, the microwave oven, drones, robots, washing machines, etc. Hence, embedded systems technology is a multidisciplinary field that is synonymous with fields like computer science, control engineering, mechatronics, robotics, etc. (7), as shown in Fig. 1.

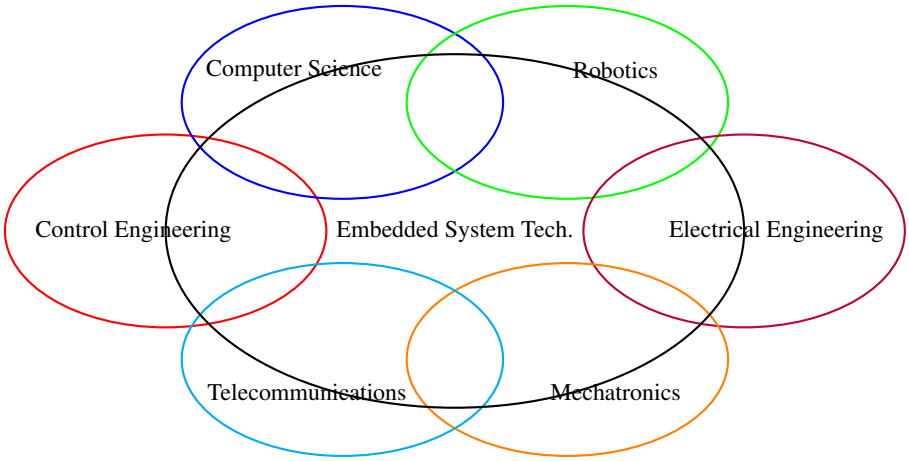


Figure 1: Diagram showing disciplines that overlap Embedded System Technology

2.1. Uses of embedded systems

Embedded systems can be used to:

1. Robotize a machine or any mechanical system.
2. Automate factories and production plants.
3. Computerize a motor vehicle or an aircraft.
4. Control electric toy cars and drones.

The applications of embedded systems cannot be over-emphasized, and with the recent boom in the field of automation and the Internet of things, embedded systems have found new technological disciplines like physical computing and cyberphysical systems (8; 9; 10). Popular embedded systems platforms include the Arduino microcontroller and Raspberry Pi single-board computer (SBC) as discussed in Section 2.3, which are the major platforms for skill-based training in the field

2.2. Benefits of the embedded Systems training

1. Considering the fact that many manufacturing industries around the World are moving from hard mechanization to automation (11), some basic knowledge of embedded systems with practical content will open up maintenance job opportunities for trainees in automated factories where they would be retooling and reprogramming production systems.
2. For conventional engineering graduates, our curricula framework would help them improve their knowledge and computational proficiency, regardless of their engineering background (12). For instance, a mechanical engineer who knows more about how to build machines would learn how to computerize (i.e., automate) them if they undergo this course prescription.
3. Artisans such as mechanics, machinists, plumbers, electricians, carpenters, and craftsmen who undergo the course prescription will essentially update their knowledge with the current trends in embedded systems technology, which will positively influence their craft.
4. As demonstrated by (13; 14), regular engineering technology students, especially those in mechatronics, computer engineering, computer science, mechanical engineering, agricultural engineering, and electrical/electronic engineering, will gain a deep understanding of the control theory they are taught in their respective classes and this is expected to positively influence the quality of their thesis in the area of robotics, avionics, automation, and wireless control of systems.

2.3. Some Embedded Systems Platform

Some commercially available microprocessor-based electronic control devices that are crucial in the development of simple or medium-scale embedded systems and automation are reviewed. These devices are classified into their categories:

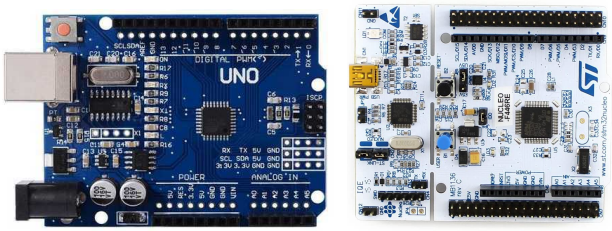
1. Microcontrollers
2. SBC or multi-core edge devices
3. Programmable logic controllers

For example, embedded programs are written to show how some of these embedded/edge controllers can be used to control a servo motor (e.g., the MG90 servo), as described in (15).

The servo motor (or simply the servo) operates on the basic principle of servomechanism and pulsed-width modulation (16). This is a mechanical feedback position control system that is widely used in robotics and automation. The MG995 servo contains a DC motor, a speed reduction gear train, an angle sensor (that is, a potentiometer for the detection of angular position), and a dedicated control circuit as shown in (15). The DC motor is attached to the gear train to control the movement of the output axis. As the motor rotates, the angle sensor's potentiometric resistance changes, so that the control circuit can precisely regulate the movement and angular position of a lever that is attached to the servo's output axis. When the lever is at the desired position, the power supply to the servo is stopped according to the PID algorithm running in the control circuit; if not, the servo is turned in the appropriate direction.

2.4. Microcontrollers

The microcontroller is a complete single-chip computer system that is optimized for the main function of control. Basically, the microcontroller comprises a microprocessor, read-only memory (ROM), random-access memory (RAM), several input/output (I/O) interfaces, and one or more serial ports. Modern microcontrollers are enhanced with higher-processor speed, Radio/Wi-Fi capability, sufficient memory, an integrated analog-digital converter (ADC), and a boot-loader (i.e., an embedded operating system)to facilitate objected-oriented programming (17). These novel features have opened the possibility of implementing Internet of Things(IoT) functions on the microcontroller. Commonly used microcontrollers in the context of IoT and CPS include the Arduino and STM microcontroller, shown in Figures 2(a) and (b), respectively.



(a) Arduino microcontroller (b) STM microcontroller

Figure 2: Frequently used microcontroller platforms for embedded systems projects

2.4.1. Arduino microcontroller

Arduino is an open-source platform that is used to develop embedded systems. It consists of both a physical programmable circuit board shown in Figure 2(a) (often referred to as a microcontroller) and a desktop-based integrated development environment (IDE). Control programs are written in this IDE and uploaded to the Arduino board. The Arduino platform has become quite popular with people just starting out with electronic controls. Unlike most previous programmable circuit boards, the Arduino does not need a separate piece of hardware (called a programmer) in order to load new code onto the board – you can simply use a universal serial bus (USB) cable. Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn how to program (18).

Arduino is a popular easy-to-use device with several peripheral-specific libraries. It is readily available in many parts of the world at affordable prices. It runs different variants of Atmel's microcontroller chip and can be programmed with a special C/C++ language. Due to its versatility, it is now founding a new application in industrial automation. Controlling a servo with the Arduino microcontroller, like many other sensors and actuators, requires a library (19; 20). In this case, servo.h library was used as given in (Listing 1). This spares the developer's time and effort in configuring a function to relate pulse width values to the various servo angles.

Listing 1: Arduino compatible C++ servo control codes

```
#include <Servo.h>

Servo myservo; // Create a servo object to control a servo
// 12 servo objects can be created on most boards.
void setup() {
  myservo.attach(9); //attach servo on pin-9 to the servo object
}

void loop() {
   myservo.write(0);       // 0 deg position
   delay(1000);

   myservo.write(90);     // 90 deg position
   delay(1000);

   myservo.write(180);   // 180 deg position
   delay(1000);
}
```

2.4.2. STM-32 microcontroller

The STM microcontroller is an affordable and embedded system. A typical variant is the STM32 family, which has 32-bit microcontroller integrated circuits that are being manufactured and sold by STMicroelectronics. At the heart of the STM microcontroller is the 32-bit ARM Cortex-M processor core. STM32 microcontrollers offer a large number of serial and parallel communication peripherals which can be interfaced with all kinds of electronic components including sensors, displays, cameras, motors, etc. All STM32 variants come with internal flash memory and RAM (21).

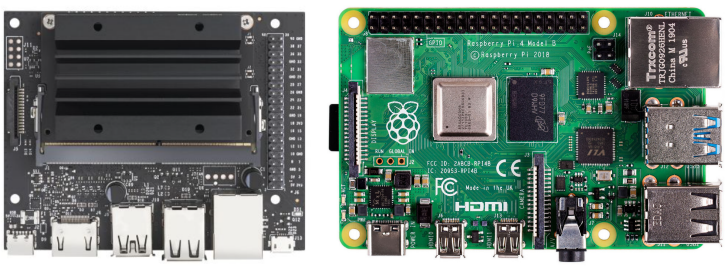
Unlike Arduino, this comes in handy when high computing power and intersystem connectivity are required for localized control in a cyber-physical system. Using the STM microcontroller for control is a little more complicated than using the Arduino and requires the configuration of an ECU abstraction layer (ECUAL) into the software architecture of the STM microcontroller. An example of an STM main source code for controlling the servo is given in (Listing 2). Just as in Arduino, this is also a modified C/C++ program.

Listing 2: STM32 compatible C++ servo control codes

```
#include "main.h"
#include "../ECUAL/SERVO/SERVO.h"

// The Servo Instance Index Must Start From 0
#define myservo 0

void SystemClock_Config(void);
static void MX_GPIO_Init(void);

int main(void)
{
    HAL_Init();
    SystemClock_Config();
    MX_GPIO_Init();
    SERVO_Init(myservo);
    while (1)
    {
        SERVO_MoveTo(myservo, 0);      // 0 deg position
        HAL_Delay(1000);
        SERVO_MoveTo(myservo, 180);  // 90 deg position
        HAL_Delay(1000);
        SERVO_MoveTo(myservo, 90);  // 180 deg position
        HAL_Delay(1000);
    }
}
```

2.5. Multi-core edge devices

This class of embedded systems comprises the SBC, also known as artificial intelligence (AI) microchips. They are multiprocessor computing platforms on a single chip that are used to execute high-performance computing in embedded systems such as robots and other automated machines. According to (17), these devices compete well with many desktop computers in terms of computing power, graphics processing, and versatility. This is attributed to the continuous shift from single-core to multicore processors in embedded systems and automation, coupled with the availability of efficient parallel programming technologies. Typical examples include the Nvidia Jetson Nano and the Raspberry Pi 4B shown in Figure 3, which become very crucial when two or more algorithms are to be executed in parallel.



(a) Nvidia Jetson Nano (b) Raspberry-Pi 4B

Figure 3: Common multi-core edge devices

With edge computing platforms, the CPS developer can now create complex edge intelligent systems that were only possible with conventional desktop computers. The concept of edge computing is also applicable to various aspects of agricultural automation and data communication (22). As an example, a simple Raspberry Pi-based Python program for controlling the MG90 servo is given in (Listing 3).

Listing 3: Raspberry-Pi compatible Python servo control codes

```
import RPi.GPIO as GPIO
from time import sleep

GPIO.setmode(GPIO.BOARD)
GPIO.setup(11, GPIO.OUT)

pwm=GPIO.PWM(11, 50)
pwm.start(0)
pwm.ChangeDutyCycle(5)      # 0 deg position
sleep(1)

pwm.ChangeDutyCycle(7.5)  # 90 deg position
sleep(1)

pwm.ChangeDutyCycle(10)  # 180 deg position
sleep(1)

pwm.stop()
GPIO.cleanup()
```

The servo control codes in (Listing 1, 2, and 3) have similar syntax and semantics, and the same algorithmic structure, but are encrypted in different embedded programming languages and for different control devices. Servomechanism and servomotors constitute indispensable technology in automation and CPS applications. These include precision steering in drive-by-wire, vehicular farm machinery, automation of food processing machines, mobile robotics, mechanical systems, and controlling active suspensions in vehicles. Hereafter, control codes describing embedded software functions will be primarily presented in algorithmic formats for brevity and space efficiency. This format allows for a concise representation of codes while maintaining clarity and facilitating easier comprehension and understanding of embedded software programs (23), making it easier to grasp the underlying logic and control functions.

2.6. Programmable logic controllers

The programmable logic controller (PLC) is an embedded system that has been ruggedized and adapted for the control of electromechanical production systems and processes, such as machine tools, assembly lines, chemical plants, industrial robots, power plants, reactors, etc. Current PLCs use at least one of the following five programming languages: Ladder Diagram, Structured Text, Function Block Diagram, Instruction List, or Sequential Function Charts. In Europe, the CODESYS standard is a common practice for the use of PLCs for system automation. Given in Algorithm 1 is an example of a PLC-based structured text program in CODESYS that uses a PID algorithm to control the speed of a DC motor. Here, the Main function contains variables for the desired speed (`DesiredSpeed`) and the current speed (`CurrentSpeed`) of the DC motor. It also includes variables for the PID controller parameters (`Kp`, `Ki`, and `Kd`), as well as other variables for the calculation of the PID. The `PIDController` method is responsible for calculating the output of the PID controller based on the error between the desired and current speed. It calculates the integral and derivative terms, limits the output to a maximum value, and applies the output to control the speed of the DC motor. This example of a PLC-based structured text program using a PID algorithm for controlling the speed of a DC motor is highly relevant to industrial automation. It demonstrates the practical implementation of PID control, which is a fundamental technique in various industrial processes. Understanding and mastering such programming concepts is crucial for students and professionals in technical education fields related to industrial automation.
Because of the rugged nature of the PLC and its capacity to operate in extremely unstable and highly dynamic conditions, it has become an attractive control platform for automating agricultural machinery and earth-moving vehicles. For instance, (24) proposed the automation steering systems for heavy farm machinery using the Siemens S7-200 intelligent PLC, to enhance the teleoperation of physical farm work. With the advent of Arduino PLC modules, such as the Controllino PLC module shown in Figure 4, which could be easily programmed with the conventional Arduino IDE and its variant of the C / C++ programming language, it is expected that the adaptation of PLCs to automation will increase in both an industrial or non-factory environment. These include emerging areas of smart agricultural technology such as vertical farming and the already mentioned auto-mechanization of physical farming. An additional impetus to this is its affordability and adaptability as an educational apparatus in schools, especially those in which robotics and Arduino programming have already been introduced. The present writing acknowledges the relevance of reviewing various controllers and their adaptability to different embedded system applications. Therefore, the next sections use models and practical examples to outline more potential areas of technology in which these devices could be employed for teaching and learning.

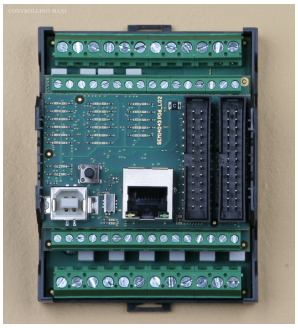


Figure 4: The Arduino-based Controllino PLC module

Algorithm 1 PID-based structured text algorithm that controls DC motor speed

```
VAR
    DesiredSpeed: REAL                                  ▷ Desired speed of the DC motor
    CurrentSpeed: REAL                                  ▷ Current speed of the DC motor
    Error: REAL                                         ▷ Difference between desired and current speed
    PreviousError: REAL                                 ▷ Error in the previous cycle
    Integral: REAL                                      ▷ Integral term of the PID controller
    Derivative: REAL                                    ▷ Derivative term of the PID controller
    Output: REAL                                        ▷ Output of the PID controller
                                                        ▷ PID controller parameters
    Kp: REAL ← 0.5                                      ▷ Proportional gain
    Ki: REAL ← 0.1                                      ▷ Integral gain
    Kd: REAL ← 0.2                                      ▷ Derivative gain
                                                        ▷ Other variables and constants
    SampleTime: TIME ← 1s                               ▷ Sampling time of the PID controller
    MaxOutput: REAL ← 100                               ▷ Maximum output value

procedure PIDCONTROLLER
        TimeDifference: TIME                            ▷ Time elapsed since the last cycle
                                                        ▷ Calculate the time difference
        TimeDifference ← TIME_TO_REAL(T - SampleTime)   ▷ T is the clock time
                                                        ▷ Calculate the error between desired and current speed
        Error ← DesiredSpeed - CurrentSpeed
                                                        ▷ Calculate the integral term
        Integral ← Integral + (Ki * Error * TimeDifference)
                                                        ▷ Calculate the derivative term
        Derivative ← (Kd * (Error - PreviousError)) / TimeDifference

                                                        ▷ Calculate the output of the PID controller
        Output ← Kp * Error + Integral + Derivative
                                                        ▷ Limit the output within the maximum value
        IF Output > MaxOutput THEN
            Output ← MaxOutput
        ELSIF Output < -MaxOutput THEN
            Output ← -MaxOutput
        END_IF
                                                        ▷ Update the previous error for the next cycle
        PreviousError ← Error
                                                        ▷ Apply the output to control the speed of the DC motor
                                                        ▷ (Code to control the motor based on the Output value)
end procedure
```

3. The Pedagogical Framework

Several curriculum and teaching plans proposed by researchers, including Kastelan (3), Ricks (4), Shen et al. (25), Rasch (26), and Seviora (27), have addressed the educational aspects of embedded systems. For instance, Kastelan put forward a unified embedded system syllabus with a focus on software engineering (3), while Ricks structured their curriculum based on the widely recognized IEEE/ACM model curriculum (28). The IEEE/ACM model curriculum is a standardized curriculum jointly developed by the Institute of Electrical and Electronics Engineers (IEEE) and the Association for Computing Machinery (ACM) (28). It is regularly updated to incorporate advancements in the field and align with industry needs. However, most of the existing curriculum proposals lack in-depth discussion and examples of course contents specifically tailored to embedded systems. This paper aims to fill that gap by providing models and practical examples to outline potential areas of technology where embedded systems, especially those based on the examined devices, and how these devices can be effectively employed for teaching and learning.

By addressing this gap, the present study contributes to the improvement of embedded systems education. Educators can benefit from the insights and resources provided to enhance their curriculum and teaching approaches in the field of embedded systems. Therefore, table 1 presents a brief overview of the courses included in the curriculum. This table provides a concise summary of the objectives, topics, and activities covered in each course. It offers a glimpse into the wide range of subjects explored throughout the program, including an introduction to embedded systems, computer hardware laboratory, sensors/actuators laboratory, basics of C++ programming, microcontroller laboratory, motion control laboratory, IoT, agricultural automation, etc., and ending with a research-oriented class project. This includes using practical case studies and algorithms to describe real-world applications of embedded systems. Hence, details of these courses are extensively discussed in the remainder of this literature.

Each course has specific objectives tailored to equip students with the necessary knowledge and skills in the field of embedded systems. The curriculum covers various aspects that include foundational topics in embedded systems, computer hardware organization, sensor and actuator technologies, programming fundamentals, microcontroller applications, motion control systems, single board computers, the Internet of Things, computational farming, wireless communication techniques, prototyping methodologies, robotics, and industrial automation. Detailed information on each course, including instructional materials and activities, is provided below.

Table 1: List of Courses

Week	S/N	Course Code	Title
Week 1	1	EST111	Introduction to Embedded Systems
	2	EST112	Computer Hardware Laboratory
	3	EST113	Sensors & Actuators Laboratory
Week 2	4	EST114	Algorithms & Flowcharts
	5	EST115	C++ Programming Basics
	6	EST115	C++ Programming Basics
Week 3	7	EST116	Microcontroller Hardware Laboratory
	8	EST117	Microcontroller Programming
Week 4	9	EST117	Microcontroller Programming
	10	EST118	Motion Control Laboratory
Week 5	11	EST210	Single Board Computers
	12	EST211	Internet of Things
	13	EST212	Agricultural Automation
Week 6	14	EST213	Wireless Communications Laboratory
	15	EST214	Prototyping Techniques
Week 7-10	16	EST215	Advanced Robotics
	17	EST216	Real-time Systems
Week 11-12	18	EST217	Industrial Automation
Week 13-15	19	EST218	Class Project

3.1. EST111 Introduction to Embedded Systems

Objectives: This lesson is aimed at introducing the students to the field of embedded systems and their applications in robotics, drones, home automation, and a range of many other physical devices.

Outline:

- Definition of embedded systems.
- Properties of systems.
- Components of an embedded system.
- Hardware components.
- Software components.
- Applications of embedded systems in robotics, automobiles, drones, home appliances, and industrial controllers.

Instructional Materials: Already existing embedded system prototypes e.g., mobile robots, drones, tele-automated farm setups, etc.
Activities: Students will be shown some prototypes of embedded systems in the workshop as examples.

3.2. EST112 Computer Hardware Laboratory

Objectives: This lesson aims to teach students about the organization of a standard computer system. Being that embedded systems are essentially computers, this course will serve as a prerequisite to understanding embedded computers which are more intricate than standard desktop computers.

Outline:

- Introduction to computer hardware.
- Microcomputer architecture.
- Components of the motherboard.
- The microprocessor – logic, architecture, and microprogramming.
- Using the parallel and serial ports for robotic control.
- Embedded computers - microcontrollers, PLCs, single-board computers (SBC).

Instructional Materials: A desktop computer, motherboard samples, microprocessor samples, Arduino microcontroller, and standard I/O devices.

Activities 1: The student will learn how to assemble a computer from its components.
Activities 2: The students will be introduced to the Arduino microcontroller and SBCs (Raspberry Pi & Lathe Panda) which are very useful for developing embedded systems.

3.3. EST113 Sensors and Actuators Laboratory

Objectives: This lesson aims at giving the students practical knowledge of sensors, actuators, and micro-electromechanical systems (MEMS) that are being used in the development of robots, drones, and many other mechatronic systems.

Outline:

- Properties of a general instrumentation system.
- Classification of sensors.
- Sensing elements – sensors and their working.
- Actuators – motors, servos, and relays.
- MEMS (i.e., Micro-machine) sensors.
- Data acquisition ICs – ADCs and DACs.
- Opt-visual monitoring devices – IP cameras, IR cameras, Laser range finder.

Instructional Materials: Temperature sensors, ultrasonic sensors, electrochemical (MQ-2) gas sensors, stepper motor, servo motors, electromagnetic (i.e., solenoid) valve, GPS module, gyroscope/accelerometer IC package (IMU), and position sensors.

Activities 1: Practicals on how to calibrate sensors.
Activities 2: Interface design for sensor and actuators with a computer, writing program statements that can read data from a smart sensor and write signals to smart actuators.

3.4. EST114 Algorithms and Flowcharts

Objectives: The objective of this lesson is to familiarize students with the fundamentals of algorithms and flowcharts, enabling them to design and analyze efficient solutions for various problems in embedded systems.

Outline:

- Introduction to algorithms and their importance in problem-solving.
- Understanding flowcharts as visual representations of algorithms.
- Characteristics of good algorithms and their types (sequential, conditional, iterative).
- Symbols and their meanings in flowcharts (start/end, process, decision, input/output).
- Designing algorithms using stepwise refinement and pseudocode.
- Mapping algorithms to flowcharts using appropriate symbols and connections.
- Flow control structures: sequential execution, conditional statements, and looping structures.
- Problem-solving techniques using algorithms and flowcharts.
- Testing, debugging, and optimizing algorithms for efficiency.
- Case studies and examples applying algorithms and flowcharts in embedded systems.

Instructional Materials:

- Visual aids displaying flowchart symbols and examples.
- Interactive flowchart software or paper-based flowchart templates.
- Embedded system case studies where algorithms and flowcharts are employed.

Activities:

- Step-by-step walkthrough of designing algorithms for simple problems.
- Collaborative creation of flowcharts for given algorithms.
- Problem-solving exercises using algorithms and flowcharts in embedded systems scenarios.
- Analysis and improvement of existing algorithms and flowcharts.
- Group discussions and presentations on real-world applications of algorithms and flowcharts in embedded systems.

3.5. EST115 C++ Programming Basics

Objectives: In this course, the student will learn how to write and run simple C++ programs using C/C++ integrated development environment (IDE). The skills accrued will be relevant to microcontrollers or embedded systems programming.

Outline:

- Overview of computer programming.
- Algorithms and pseudo codes.
- General features of C++.
- Input/output.
- Control structures – sequence, selection (branch), iteration (looping).
- Library functions.
- Arrays and strings.
- Bitwise operations.

Instructional Materials: Desktop computers running C++ IDE such as the one shown in Fig. 5, where it was been used to practice socket programming.

Activities: Students will do actual coding on a microcomputer of all the sample algorithms during the lesson. Important aspects of algorithmics, such as the use of functions and `global/local variables`, that frequently show up in embedded control algorithms could be explained. For example, Algorithm 2 showcases the difference between local and global variables in an embedded systems context. The main procedure begins by calling the `fuseHumidityData` procedure. Inside the `fuseHumidityData` procedure, local variables are

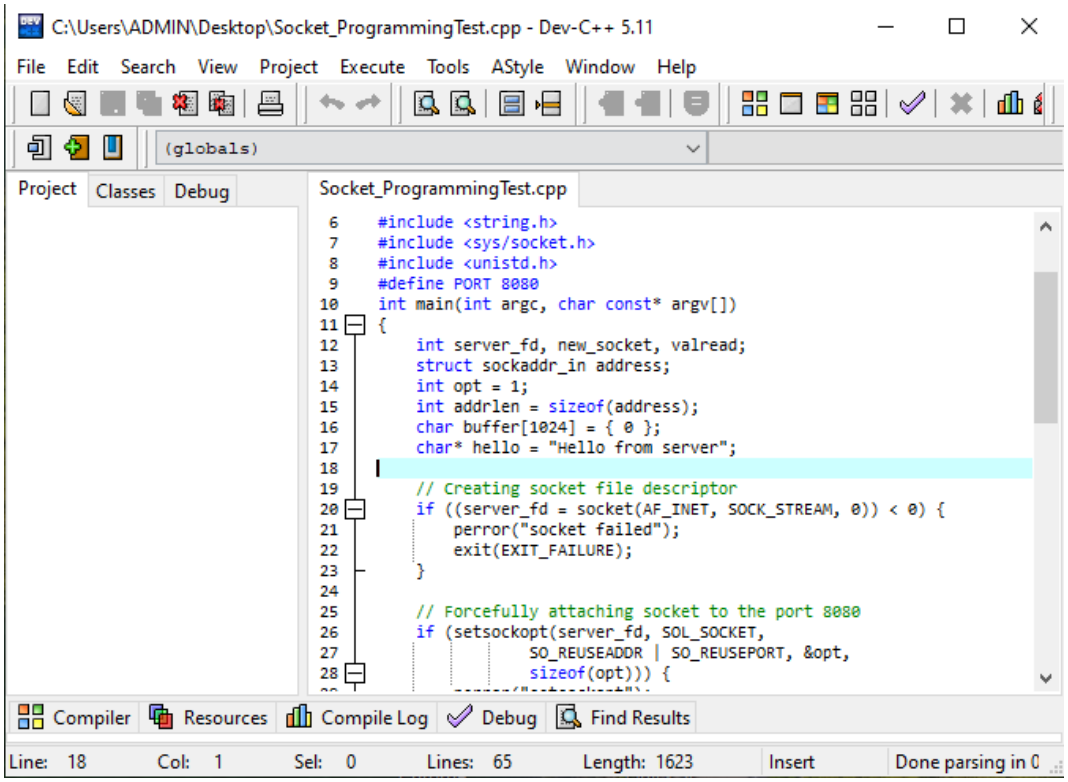


Figure 5: C++ integrated development environment

declared to store the sensor data and weights. The algorithm assumes that the sensor readings have been obtained from external sources. The sensor data are then assigned to the `localSensor1Value` and `localSensor2Value` variables.

Algorithm 2 Demonstrating the difference between local and global variables

```
Global Variables:
    float fusedHumidity
procedure FUSEHUMIDITYDATA
    Local Variables:
        float localSensor1Value
        float localSensor2Value
        float weightSensor1 = 0.6
        float weightSensor2 = 0.4
                                          ▷ Read sensor data and assign values to local variables
    localSensor1Value ← Read sensor 1 data
    localSensor2Value ← Read sensor 2 data
                                          ▷ Fusing the data using weighted average
    fusedHumidity ← (localSensor1Value × weightSensor1) + (localSensor2Value × weightSensor2)
end procedure
procedure MAIN
    fuseHumidityData()
    Output: "Fused Humidity: " + fusedHumidity
end procedure
```

The algorithm proceeds to fuse the data using a weighted average calculation. The `fusedHumidity` variable is computed by multiplying `localSensor1Value` with `weightSensor1` and adding it to the product of `localSensor2Value` and `weightSensor2`. Once the fusion process is completed, the main procedure outputs the result by displaying the message "Fused Humidity: " followed by the value of the `fusedHumidity` variable. This algorithm highlights how `local` and `global variables` are used to store and manipulate data within a program, emphasizing the importance of careful variable management and algorithmic processing in embedded software development to achieve desired outcomes and functionalities.

3.6. EST115 Microcontroller Hardware Laboratory

Objectives: In this lesson, students will learn about microcontroller hardware in general with an emphasis on Arduino microcontrollers as well as designing Arduino-based printed circuit boards (PCB). This course is essential because microcontrollers serve as electronic brains in embedded systems.

Outline:

- Overview of the microcontroller.

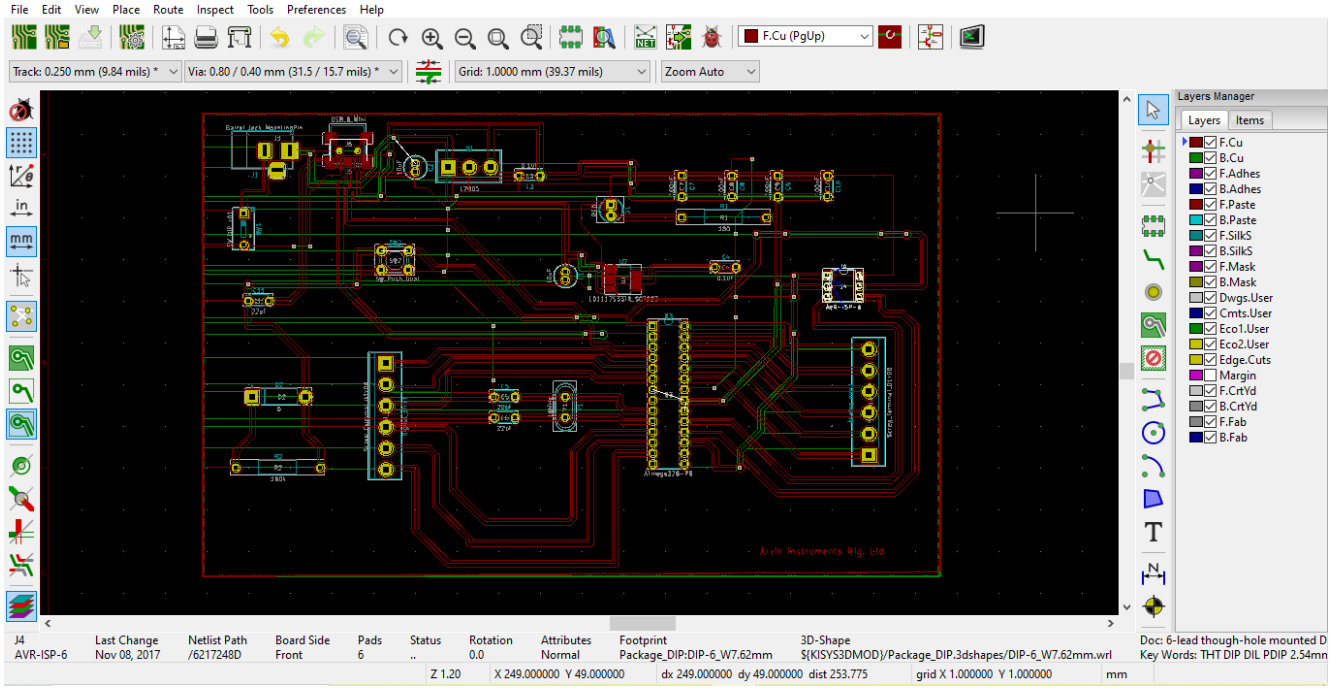


Figure 6: KiCAD integrated development environment

- Architecture of various microcontrollers(29; 30).
- Architecture of the Arduino microcontroller.
- Designing custom Arduino-based PCB with electronic design automation (EDA) tools.
- Arduino IDE and libraries.
- How to program the Arduino board using a microcomputer.
- Applications of microcontrollers in embedded systems.

Instructional Materials: Each student will buy his/her own Arduino microcontroller board but will be provided with an Arduino IDE for free which they may install on their laptops. For EDA projects, the KiCAD tool and relevant electronic prototyping materials are required.

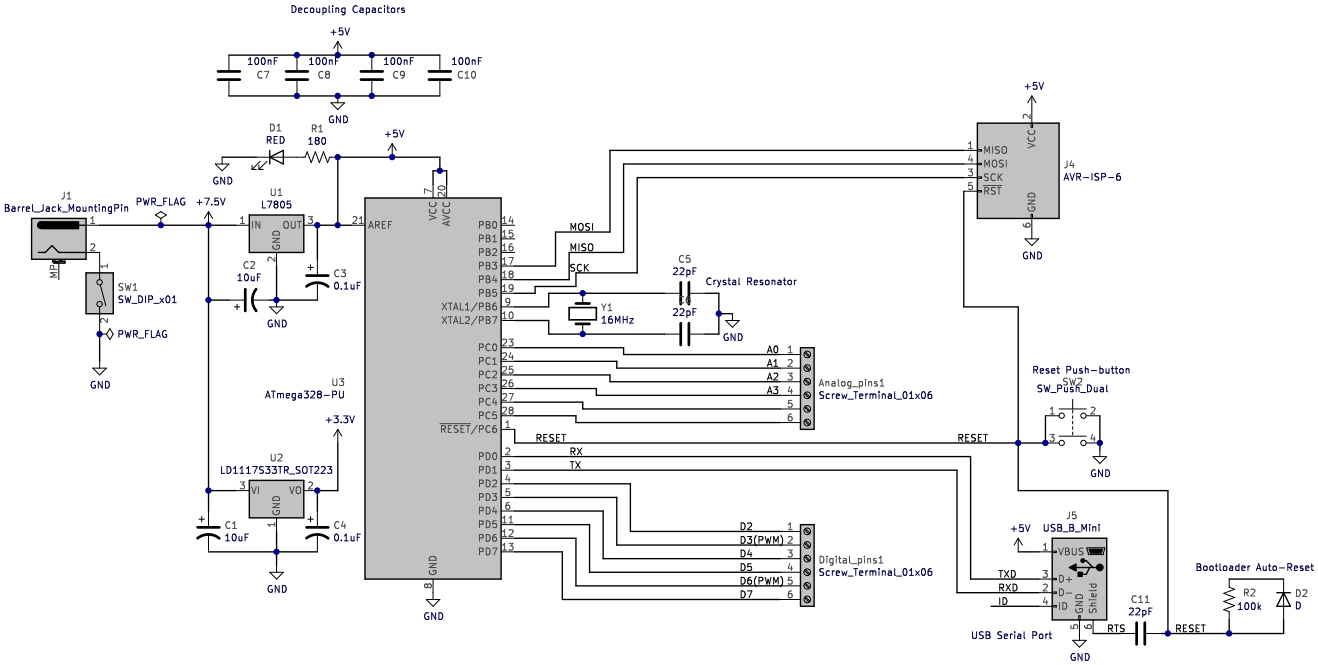
Activity: Designing custom Arduino PCB
The use of EDA tools is essential for designing, implementing, and teaching embedded systems due to the critical role they play in integrating hardware/software components and automating design procedures (31; 32). In relation to STEM education, tools like KiCAD, an open-source EDA tool, are invaluable. This tool facilitates schematic design, enabling students to visualize component connections and interactions. It also supports PCB layout, teaching students how to arrange components and route traces effectively. Through KiCAD's simulation capabilities, students can virtually test their designs, identifying potential issues before physical implementation (33). KiCAD's component library aids in understanding diverse electronic components vital for embedded systems. Also, its open-source nature ensures accessibility and encourages students to explore design concepts without financial barriers. To demonstrate these functionalities and benefits, a custom Arduino-based PCB was designed using the ATmega328P microcontroller chip (a part of the AVR microcontroller family developed by Atmel (34)). Using the KiCAD IDE, as shown in Fig. 6, the design process started with the sketching of the circuit schematic. In this phase, the required electronic components are selected and routed to relevant pin-outs as shown in Fig. 7(a). These essential components include the AVR programmer chip, a crystal resonator (for precise clock synchronization), the L7805 chip (for voltage regulation), a variety of capacitors or capacitor networks (for smoothing signals and decoupling power sources), a bootloader, and various connectors/switches.

In the next phase, the circuit diagram was transformed into a PCB floor plan. This served as the blueprint for crucial tasks like routing buses, placing vias, and selecting materials and layers as shown in Fig. 7(b). During this phase, specific networks were highlighted for clarity, fill zones were added to optimize power distribution, keep-out areas were defined to prevent interference, and graphic lines and text were used to visually organize the design. Board dimensions were also set to ensure the design fits within the available space. The final form of the PCB design is depicted in Fig. 7(c). This design was carefully validated using KiCAD's electrical rules checker prior to fabrication.
Replicating this exercise within microcontroller hardware laboratories imparts the skill of resource optimization for electronic prototyping in embedded systems projects that are tailored to specific application needs. This approach ensures that only the required microcontroller pins and peripherals are utilized. By mastering EDA tools like KiCAD, students gain practical skills, which prepare them for real-world engineering challenges in embedded systems development. This hands-on experience bridges theoretical knowledge with practical application, empowering students to confidently navigate the complexities of designing embedded systems for various applications.

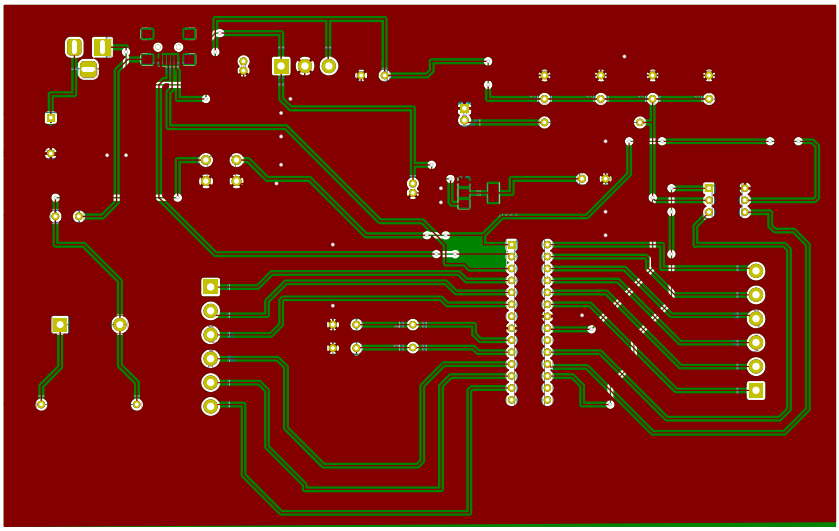
3.7. EST117 Microcontroller Programming

Objectives: The aim of this lesson is to train students on how to program an Arduino microcontroller for mechatronics and robotics projects.

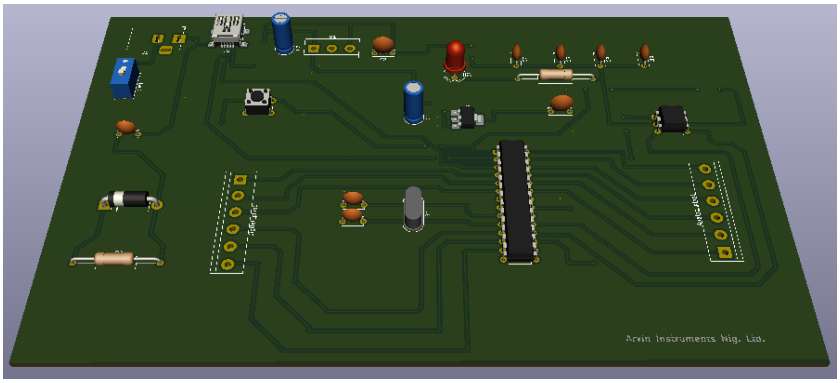
Outline:



(a) Circuit diagram



(b) 2D layout of PCB (Red: Cu plating, Green: Exposed substrate material.)



(c) 3D rendering of fully integrated PCB

Figure 7: Design of custom Arduino PCB board

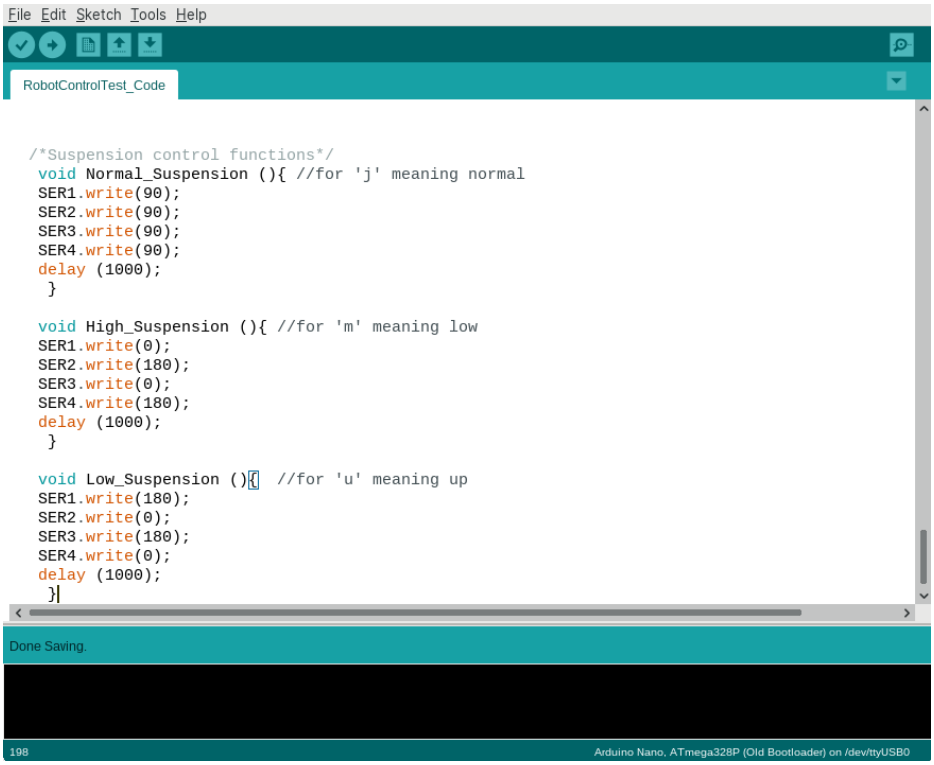


Figure 8: Arduino integrated development environment

- The Arduino IDE.
- Structure – setup(), loop(), functions, curly braces, (;) semicolon.
- Variables.
- Data types.
- Arithmetic and logical operations.
- Constants – constants, high/low, input/output.
- Digital I/Os and Analog I/Os.
- Flow control – sequence, selection, and looping.
- Timing.
- Math.
- Random.
- Serial communication.

Instructional Materials: A desktop computer, an Arduino microcontroller board, a USB cable, and an Arduino IDE.

Activities: Students will do hands-on coding of all sample algorithms with a desktop computer through the Arduino IDE, as shown in Fig. 8, and they will also test the communication and control of sensors and actuators with/by the microcontroller, respectively.

3.8. EST118 Motion Control Laboratory

Objectives: In this lesson, the student will learn about servomechanism and how to use specific mechatronic components to control the movement of mechanical links or systems servo mechanically, as described earlier in Listing 1 - 3 and Algorithm 1.

Outline:

- Introduction to servomechanism.
- Preview feedback control systems.
- Common components for power/speed/position control in robotic systems:
  1. Relays (35).
  2. Solenoid valve (for flow control).
  3. Electronic speed controller (ESC).
  4. Stepper motors.
  5. Cascades transistor circuits.
  6. H-bridge circuits e.g. L293D IC motor driver.

Activities 1: The students will learn how to interface a servomechanism or any components for motion control with the microcontroller.

Activities 2: The student will learn to use 5 volts of electricity to control 250 volts of electricity (this skill will be useful in the practice of industrial or home automation).

Activities 3: The students will learn how to write programs that run on a microcontroller for the operation of relays, IC motor drivers, servo motors, and stepper motors.

3.9. EST210 Single Board Computers

Objectives: The aim of this lesson is to familiarize the students with common single-board computers (SBCs) which are small computing hardware for implementing the Internet of Things.

Outline:

- Introduction to SBCs.
- Some commercially available SBCs.
- Introduction to Linux operating system.
- The Raspberry Pi – Architecture and functions.
- Overview of the Windows 10 operating system.
- The Lathe Panda – Architecture and functions.
- Connecting external peripheral to SBCs.
- Applications of SBCs in the development of cyberphysical systems and home automation.

Instructional Materials: A Raspberry Pi SBC and a desktop computer.

Activities: The students will practice how to interface sensors, actuators, modems, and I/O devices to an SBC. These skills will come in handy during the design of cloud robots and farm automation.

3.10. EST211 Internet of Things (IoTs)

Objectives: The aim of this lesson is to train the students on how to create networked embedded systems that can be controlled via Internet connections. The application of this system, especially in agriculture, will be explored extensively.

Outline:

- Definition of a cyberphysical system.
- Components of a generic cyberphysical system.
- Overview of computer networking and the Internet.
- Introduction to remote computing.
- Introduction to terminal simulators.
- The Internet of Things (IoTs) (36).
- Putting it all together.
- Creating a simple cyberphysical system with SBCs and the IoTs.
- Applications in robotics (37; 10; 38), home automation (39), and agriculture (40).

Activities: Students will be trained in various methods of using SBCs and the Internet to control motors, actuators, and servo systems using the Internet instead of the ordinary radio frequency. A foundational aspect of this area to be taught to young enthusiasts is the concept of socket programming – a method that enables communication between computers over a network by establishing a connection using sockets. This allows data to be sent and received between client and server applications, facilitating network-based communication and data exchange (41; 42; 43), making it applicable in areas such as telemedicine to facilitate secure and efficient transmission of medical data between healthcare professionals and patients as well as controlling surgical robots, industrial IoT, and vehicle telematics.

Algorithm 3 and Algorithm 4, for example, describes the format of client and server Python scripts utilizing socket programming to enable remote control of a relay's logical state over the Internet. The relay is connected to a Raspberry Pi Single Board Computer (SBC) acting as the server, while the client program on a PC communicates with the server over the Internet to control the relay. The client program described in Algorithm 3 establishes a socket connection with the server program described in Algorithm 4 using the server's IP address and port number. It prompts the user for input to control the relay, sending the corresponding command ('ON' or 'OFF') to the server. The server program, running on the Raspberry Pi, listens for client connections, accepts incoming requests, and handles them accordingly. Upon receiving a command from the client, the server activates or deactivates the relay connected to a GPIO pin on the Raspberry Pi using the `RPi.GPIO` library. It provides a simple interface to control the relay by setting the GPIO pin to high or low.
This client-server architecture, as modeled in Fig. 9, allows for seamless communication between the PC and Raspberry Pi, enabling relay control remotely. Hence, the relay can be used to switch various electrical devices or control other components in a larger system. Socket programming provides a flexible and efficient means of inter-device communication over a network. It opens up possibilities for integrating and controlling different devices in a distributed environment. Additionally, the use of the Raspberry Pi as the server adds the advantage of its GPIO capabilities, enabling the direct control of physical devices.

By understanding and adapting this example, students can develop more sophisticated cyberphysical systems/IoT applications where remote relay control is a requirement. In essence, the concept of socket programming can be used by teachers to showcase the potential and versatility of embedded systems like the Raspberry Pi in enabling remote control and automation.

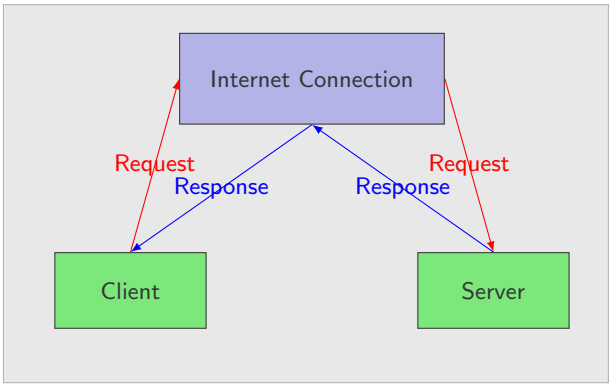


Figure 9: Block diagram of Client-Server Architecture

Algorithm 3 Socket-based client algorithm to control a relay

Require: `socket`, `SERVER_IP`, `SERVER_PORT`, `COMMAND_ON`, `COMMAND_OFF`
Ensure: Relay control commands are sent to the server

```
client_socket ← socket.socket(socket.AF_INET, socket.SOCK_STREAM)
procedure CONNECTTOSERVER
    Connect to the server:
        client_socket.connect((SERVER_IP, SERVER_PORT))
        print("Connected to the server.")
end procedure
procedure RELAYCONTROLLOOP
    while True do
        User Input:
            user_input ← input("Enter 'on' to turn on the relay or 'off' to turn it off:  ")
        if user_input.lower() == 'on' then
            Send command to server:
                client_socket.sendall(COMMAND_ON)
        else if user_input.lower() == 'off' then
            Send command to server:
                client_socket.sendall(COMMAND_OFF)
        else
            Invalid command:
                print("Invalid command.  Please try again.")
        end if
    end while
end procedure
procedure HANDLECONNECTIONREFUSED
    Failed to connect to the server:
        print("Failed to connect to the server.")
end procedure
procedure CLOSECONNECTION
    Close the socket:
        client_socket.close()
end procedure
Procedure Calls:
    connectToServer()
    relayControlLoop()
    handleConnectionRefused()
    closeConnection()
```

Algorithm 4 Socket-based server algorithm to control a relay

```
Require: socket, RPi.GPIO
SERVER_IP ← '0.0.0.0'
SERVER_PORT ← 1234
RELAY_PIN ← 18
server_socket ← socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind((SERVER_IP, SERVER_PORT))
GPIO.setmode(GPIO.BCM)
GPIO.setup(RELAY_PIN, GPIO.OUT, initial=GPIO.LOW)
procedure HANDLECLIENT(client_socket)
    while True do
        Receive command from client:
            command ← client_socket.recv(1024)
        if command == b'ON' then
            Turn on the relay:
                GPIO.output(RELAY_PIN, GPIO.HIGH)
                print("Relay turned on.")
        else if command == b'OFF' then
            Turn off the relay:
                GPIO.output(RELAY_PIN, GPIO.LOW)
                print("Relay turned off.")
        end if
    end while
end procedure
Server Setup:
    server_socket.listen(1)
    print("Waiting for client connections...")
Main Server Loop:
while True do
    Accept client connection:
        client_socket, client_address ← server_socket.accept()
        print("Connected with client:", client_address)
        handleClient(client_socket)
end while
procedure SERVERINTERRUPTED
    Handle server interruption:
        print("Server interrupted.")
end procedure
procedure CLEANUPANDCLOSE
    Clean up GPIO and close server socket:
        GPIO.cleanup()
        server_socket.close()
end procedure
Procedure Calls:
    serverInterrupted()
    cleanupAndClose()
```

3.11. EST212 Agricultural Automation

Objectives: The aim of this lesson is to expose the students to the practical application of robotics technology in agriculture. They will learn how to use the computer to remote control farm implements.
Outline:

- Agricultural automation – opportunity, challenges, and prospects.
- Some examples of agricultural robots in existence.
- The farm of the future.
- Construction of internet-operated poultry, as idealized in (40).
- Economics of computer/internet-assisted farming.

Activities: The program can serve as a catalyst for continuous improvement and development of telerobotic agriculture, especially in fish and poultry farming. Students should experience remote farming from a desktop computer, as discussed in (40; 44).
Inspired by the work of Pramanik *et. al.*(45) for example, the code in Algorithm 5 is an Arduino-based pseudocode that controls the irrigation of a smart farm using a water pump and a soil humidity sensor. The program continuously monitors the soil moisture level through the sensor. When the moisture level drops below a certain threshold, the program activates the water pump to irrigate the farm until the desired moisture level is reached. This automated control program can ensure efficient water usage and maintains optimal conditions for plant growth in the smart farm. In the context of a smart environment, this irrigation control framework holds significant potential for automating large-scale hydrological systems. By applying this framework, it becomes possible to efficiently pump water from riverine areas to desert-encroached hinterlands.

This automation system can play a crucial role in addressing deforestation problems and ensuring the sustainable growth of vegetation in arid regions, as suggested in (46; 47). Through intelligent management of water resources, this framework contributes to the preservation of natural ecosystems and promotes environmental balance.

Algorithm 5 Arduino-based algorithm that controls water irrigation water pump based on soil humidity measurement

Require: Arduino

```
                                                          ▷ Pin connections
const int soilHumidityPin ← A0
const int waterPumpPin ← 9
                                                          ▷ Threshold for soil moisture level
const int moistureThreshold ← 500

procedure SETUP
                                                          ▷ Initialize serial communication
    InitializeSerial(9600)
                                                          ▷ Set water pump pin as an output
    SetPinMode(waterPumpPin, OUTPUT)
end procedure
procedure LOOP
                                                          ▷ Read value from soil humidity sensor
    int soilHumidity ← analogRead(soilHumidityPin)
                                                          ▷ Print soil humidity value to serial monitor
    Print("Soil Humidity: ")
    Println(soilHumidity)
                                                          ▷ Check if soil moisture is below the threshold
    if soilHumidity < moistureThreshold then
                                                          ▷ Turn on the water pump
        digitalWrite(waterPumpPin, HIGH)
        Println("Water Pump ON")
    else
                                                          ▷ Turn off the water pump
        digitalWrite(waterPumpPin, LOW)
        Println("Water Pump OFF")
    end if
                                                          ▷ Delay before the next reading
    delay(1000)
end procedure
```

3.12. EST213 Wireless Communications Laboratory

Objectives: The aim of this lesson is to train students on how to control mechatronic systems remotely using wireless communication protocols as described in (48). This can be accomplished using XBee or NRF24L01 modules for mid-range communication and general packet radio service (GPRS) or the newer 4G/5G network for long-range communication (49).
Outline:

- Introduction to wireless communication.
- XBee/NRF24L01 radio frequency modules.
- Internet access via GPRS.
- Integration of XBee modules with Arduino microcontroller.
- Integration of GPRS(50) or 4G/5G network with Arduino microcontroller(51).
- Writing program statements to control embedded systems over a wireless network.

Activities: Students will be trained on how to interface wireless modules with embedded systems to control various mechatronic systems. For example, Algorithm 6 and Algorithm 7 are pseudocodes that describe Arduino programs, based on NRF24L01 radio communication front-end devices, which transmit three agricultural sensor data at once.
Algorithm 6 acts as a collection box, facilitating the wireless transmission of sensor data in agricultural sensing. It initializes the communication parameters and structures the sensor data for humidity, temperature, and light intensity. The algorithm packages the data into packets and transmits it as a whole using the radio module. Controlled delays between transmissions ensure efficient data transfer. At the receiver end, Algorithm 7 comes into play. Once a data packet is detected and becomes available, the algorithm reads the received sensor data and extracts the specific sensor's value for humidity, temperature, and light intensity, before printing them out as output. By efficiently receiving and processing the transmitted sensor data, Algorithm 7 facilitates real-time monitoring. As demonstrated by (52) and (53), this technique can serve as a crucial component of agricultural automation systems for harnessing farm multisensor data to gain valuable insights on how to optimize agricultural processes.

3.13. EST214 Prototyping Technique

Objectives: This lesson aims to expose students to some of the methods and tools involved in the development of an embedded system.
Outline:

- Basics of computer-aided design of systems.
- Drafting of structures using AutoCAD.
- Design of electronic circuits using Proteus.
- PCB prototyping.
- AutoCAD and 3D Printing.
- Fabrication of physical and mechanical parts using machine tools.

Activities 1: Students will do hands-on computer-aided design of robot mechanical parts and electronic circuits using AutoCAD and Proteus, respectively.

Algorithm 6 Wireless Sensor Data Transmission

Require: `SPI.h` and `RF24.h` libraries

```
RF24 radio(9, 10)                                              ▷ CE, CSN pins
struct SensorData {
  float humidity;
  float temperature;
  int lightIntensity;
};
SensorData sensorData;
procedure SETUP
   radio.begin();
   radio.openWritingPipe(0xF0F0F0F0E1LL)        ▷ Set the address to which the data will be transmitted
end procedure
procedure LOOP                                                 ▷ Read sensor data here
   sensorData.humidity ← analogRead(A0);          ▷ Read humidity sensor value from analog input pin A0
   sensorData.temperature ← analogRead(A1);       ▷ Read temperature sensor value from analog input pin A1
   sensorData.lightIntensity ← analogRead(A2);    ▷ Read light intensity sensor value from analog input pin A2
   radio.write(&sensorData, sizeof(sensorData));                ▷ Transmit sensor data wirelessly
   delay(1000);                                                 ▷ Delay between transmissions
end procedure
```

Algorithm 7 Wireless Sensor Data Reception

Require: `SPI.h` and `RF24.h` libraries

```
RF24 radio(9, 10)                                              ▷ CE, CSN pins
struct SensorData {
  float humidity;
  float temperature;
  int lightIntensity;
};
SensorData sensorData;
procedure SETUP
   Serial.begin(9600);
   radio.begin();
   radio.openReadingPipe(1, 0xF0F0F0F0E1LL)     ▷ Set the address to which the data is transmitted
   radio.startListening();
end procedure
procedure LOOP
   if radio.available() then
      radio.read(&sensorData, sizeof(sensorData));
      float humidity ← sensorData.humidity;                    ▷ Process received sensor data
      float temperature ← sensorData.temperature;
      int lightIntensity ← sensorData.lightIntensity;
      Serial.print("Humidity:  ");                             ▷ Print sensor data
      Serial.print(humidity);
      Serial.print("Temperature:  ");
      Serial.print(temperature);
      Serial.print("Light Intensity:  ");
      Serial.println(lightIntensity);
   end if
end procedure
```

Activities 2: Students will experience how a printed circuit board (PCB) is being realized by the chemical etching method.
Activities 3: Students will do hands-on fabrication of specified physical and/or mechanical components using the relevant machine tools.

3.14. EST215 Advanced Robotics

Objectives: The aim of this lesson is to provide advanced knowledge and skills in the field of robotics, including advanced control algorithms, computer vision, and machine learning for robotics applications.
Outline:

- Review of robotics fundamentals.
- Advanced control algorithms for robot motion planning and trajectory generation.
- Computer vision techniques for robot perception and object recognition.
- Machine learning for robotics applications, including reinforcement learning and neural networks.
- Integration of advanced robotics algorithms and techniques into practical robotic systems.

Instructional Materials: Robotic kits, computer vision software, and machine learning frameworks.
Activities: Students will work on advanced robotics projects that involve implementing and testing advanced control algorithms, computer vision techniques, and machine learning algorithms in real robotic systems.

3.15. EST216 Real-time Systems

Objectives: The aim of this lesson is to introduce students to real-time systems and their applications in embedded systems and robotics.
Outline:

- Introduction to real-time systems.
- Real-time operating systems (RTOS).
- Task scheduling and synchronization in real-time systems.
- Real-time communication protocols.
- Real-time system design and analysis.
- Case studies of real-time systems in embedded systems and robotics.

Instructional Materials: Real-time operating systems, microcontrollers, and real-time communication modules.
Activities: Students will design and implement real-time systems using microcontrollers and real-time operating systems. They will also analyze and evaluate the real-time performance of their systems. For example, Algorithm 8, describes a face recognition algorithm that uses an identification database to identify persons in video captures.
The algorithm employs facial recognition to identify individuals captured in a video stream. It loads an identification database containing images, names, nationalities, and dates of birth. By comparing face encodings, it determines matches between the captured faces and database entries. If a match is found, the algorithm prints the person's name, nationality, date of birth, and contacts. Bounding boxes are drawn around the detected faces in the video feed for visualization. The algorithm continuously processes frames from the video stream until interrupted, allowing for real-time facial recognition and information retrieval.

Algorithm 8 Facial Recognition using Identification Database

```
Load the identification database                ▷ Each entry contains image, name, nationality, date of birth, and contact
Initialize the video camera
while True do
    Capture frame from the video camera
    Detect faces in the frame
    for all detected faces do
        Extract the face encoding
        for all person in identification database do
            Compare the face encoding with the encoding in the database
            if there is a match then                ▷ Prints the person's information
                    Print "Name:", person["name"]
                    Print "Nationality:", person["nationality"]
                    Print "Date of Birth:", person["date_of_birth"]
                    Print "Contact:", person["contact"]
            end if
        end for
    end for
    Display the frame with bounding boxes around the detected faces
    Display the frame on the screen
    Check for the 'q' key press to exit the program
end while
Release the video capture and exit
```

To implement the algorithm as Python scripts on the NVIDIA Jetson AI computing platform, the Jetson platform must be configured with the necessary software, including Python, OpenCV, and the `face_recognition` library, and the video camera must is connected to the appropriate port. Algorithm 8 should be converted to an equivalent Python script incorporating the required libraries. To ensure the smooth performance of this system, the Jetson platform must be configured with the required dependencies and packages, and the camera settings and data paths must be adjusted accordingly. A promising application of this algorithmic system is that it can be implemented on a mobile robotic system and used to intelligently monitor remote locations.

3.16. EST217 Industrial Automation

Objectives: The aim of this lesson is to provide knowledge and skills in industrial automation, including programmable logic controllers (PLC), human-machine interface (HMI), and industrial communication protocols, as highlighted in (54).
Outline:

- Introduction to industrial automation.
- Basics of PLC programming (e.g. ladder diagram, structured text, Codesys framework, etc.).
- PLC hardware and architecture.
- HMI design and implementation.
- Industrial communication protocols (e.g., CANbus, CANopen, Modbus, Profibus (55)).
- Integration of PLCs, HMIs, and industrial communication systems.

Instructional Materials: PLC systems, HMI, industrial communication modules.
Activities: Students will work on industrial automation projects that involve programming PLCs, designing HMIs, and configuring industrial communication systems. They will develop and implement automation solutions for specific industrial scenarios.

3.17. EST218 Class Project (Two weeks exercise)

Objective: At this stage of the training program, the students and the instructors will design and implement an embedded system that could be part of the workshop prototypes for future exhibitions and instruction.
The project focus may be on any of the following:

- Development of an unmanned aerial vehicle.
- Development of a mobile ground robot.
- Development of an automated farm system.
- Development of a power control/energy management system.

4. Conclusion

The pedagogical framework aims to provide students with a comprehensive education in embedded systems technology. It features a structured progression of courses, starting with an introduction to embedded systems and computer hardware before moving on to the basics of software programming. This foundational knowledge serves as a prerequisite for understanding the more intricate aspects of embedded systems, such as microcontrollers, sensors, and actuators. Using the curriculum, students can explore programming languages and platforms commonly used in the development of embedded systems, such as C++, Arduino, and Python. They can learn how to program microcontrollers to communicate with and control various sensors and actuators in the real world. Hands-on activities and projects provide students with experience in computer assembly, electronic circuit prototyping, EDA/KiCAD tools, software integration, and real-time systems. Hence, students can learn about real-time operating systems, task scheduling, and synchronization, as well as real-time communication protocols, gaining insight into designing and analyzing real-time systems, while considering factors such as timing constraints and responsiveness. Furthermore, the curriculum addresses industrial automation, introducing students to PLC, man-machine interface, and industrial communication protocols. Students gain an understanding of how these components are used to automate industrial processes and develop automation solutions for various scenarios.

This pedagogical framework and curriculum (including the suggested course codes – EST111 to EST218) serve as a general guideline that can be modified according to the needs and resources of an educational institution. This can be tailored to the institution's goals, infrastructure, and expertise. Adaptations may include adding more specialized topics (e.g. edge computing, application-specific integrated circuits, system-on-chip), research/industry collaborations, or adjustments based on the level and duration of the program. Overall, the curriculum aims to equip students with the necessary knowledge and skills to excel in the field of embedded systems and their various applications, thereby providing a solid foundation that prepares them for careers in various industries, including robotics, automation, IoT, etc.

Acknowledgement

The presented curriculum was initially proposed in 2017 at the former Lagos State Polytechnic's Laspotech-ICA Aviation Centre, Ikorodu Lagos, Nigeria, as part of its short vocational training program in the area of robotics and drone Technology.

References

[1] A. E. Eiben, "Evolving robot software and hardware," in *Proceedings of the IEEE/ACM 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, (New York, NY, USA), ACM, June 2020.

[2] S. Pasricha, "Embedded systems education in the 2020s: Challenges, reflections, and future directions," in *Proceedings of the Great Lakes Symposium on VLSI 2022*, (New York, NY, USA), ACM, June 2022.

[3] I. Kastelan and M. Temerinac, "A curriculum for unified embedded engineering education," in *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 818–823, 2016.

[4] K. G. Ricks, D. J. Jackson, and W. A. Stapleton, "An embedded systems curriculum based on the ieee/acm model curriculum," *IEEE Transactions on Education*, vol. 51, no. 2, pp. 262–270, 2008.

[5] M. Barr, "Embedded systems glossary." Neutrino Technical Library. https://barrgroup.com/Embedded-Systems/Glossary, April 2007.

[6] S. Heath, *Embedded Systems Design*. EDN Series for Design Engineers, Oxford ; Boston: Newnes, 2 ed., 2003. An embedded system is a microprocessor-based system that is built to control a function or a range of functions.

[7] A. Padhyegurjar, "A guide to a career in embedded systems." https://www.electronicsforu.com/career/guide-to-career-in-embedded-systems, 2022.

[8] R. E. Achatz, "From embedded systems to cyber-physical systems: Research challenges and application areas," in *2011 IEEE 35th Annual Computer Software and Applications Conference*, pp. 1–1, 2011.

[9] C. Ebert and J. Salecker, "Guest editors' introduction: Embedded software technologies and trends," *IEEE Softw.*, vol. 26, pp. 14–18, May 2009.

[10] O. Olakanmi and M. Benyeogor, "Internet based tele-autonomous vehicle system with beyond line-of-sight capability for remote sensing and monitoring," *Internet of Things*, vol. 5, pp. 97–115, 2019.

[11] S. Team, "Industrial automation in 2020: From mechanization to automation." https://blog.spatial.com/industrial-automation-2020, April 2020.

[12] J. Ariza, "Improving embedded programming skills through physical computing activities in engineering education: A course experience," in *2022 International Symposium on Accreditation of Engineering and Computing Education (ICACIT)*, pp. 1–6, 2022.

[13] M. S. Benyeogor, O. O. Olakanmi, K. P. Nnoli, and M. C. Gwani, "Edu-Rover: Application of Unmanned Vehicle Systems for Robotics and STEM Education in Nigeria," *EAI Endorsed Transactions on Creative Technologies*, vol. 8, 2 2021.

[14] M. S. Benyeogor, A. C. Amaechi, A. A. Dahiru, O. O. Olakanmi, A. Akintola, and S. B. Okoli, "Prototyping and conceptualizing electric model vehicles to enhance automotive STEM education: Towards sustainable e-mobility," in *the 2023 IEEE German Education Conference (GECon)*, (Berlin, Germany), 2023. Under review.

[15] R. Elliott, "Hobby servos." https://sound-au.com/articles/servos.htm, Jan. 2018. Published January 2018. Accessed: 2023-6-16.

[16] N. Pinckney, "Pulse-width modulation for microcontroller servo control," *IEEE Potentials*, vol. 25, no. 1, pp. 27–29, 2006.

[17] M. S. Benyeogor, K. P. Nnoli, O. O. Olakanmi, O. I. Lawal, E. J. Gratton, S. Kumar, K. A. Akpado, and P. Saha, "An algorithmic approach to adapting edge-based devices for autonomous robotic navigation," *EAI Endorsed Transactions on Context-aware Systems and Applications*, 8 2021.

[18] A. A. Galadima, "Arduino as a learning tool," in *2014 11th International Conference on Electronics, Computer and Computation (ICECCO)*, pp. 1–4, 2014.

[19] M. Margolis, B. Jepson, and N. R. Weldin, *Arduino cookbook*. Sebastopol, CA: O'Reilly Media, 3 ed., May 2020.

[20] M. Lineros, B. Bastías, F. Muñoz, K. Aravena, M. Figueroa, L. Rodríguez, B. Villegas, F. Hinojosa, B. Gutiérrez, A. Guerra, O. Bernales, C. Román, S. Collantes, C. Vidal, and R. Villarroel, "Electronics for everybody: student practical experiences using arduino," in *2018 37th International Conference of the Chilean Computer Science Society (SCCC)*, pp. 1–8, 2018.

[21] B. Deng, Z. Bo, Y. Jia, Z. Gao, and Z. Liu, "Research on stm32 development board based on arm cortex-m3," in *2020 IEEE 2nd International Conference on Civil Aviation Safety and Information Technology (ICCASIT*, pp. 266–272, 2020.

[22] M. O'Grady, D. Langton, and G. O'Hare, "Edge computing: A tractable model for smart agriculture?," *Artificial Intelligence in Agriculture*, vol. 3, pp. 42–51, 2019.

[23] T. Gillespie, "The Relevance of Algorithms," in *Media Technologies: Essays on Communication, Materiality, and Society*, The MIT Press, 02 2014.

[24] Y. Li, W. Deng, X. Cheng, W. Song, Z. Li, and X. Zhang, "Design of electric steering wheel for agricultural machinery," *Journal of Physics: Conference Series*, vol. 1601, p. 022041, jul 2020.

[25] G. Shen, *An Embedded System Curriculum for Undergraduate Software Engineering Program*, pp. 219–232. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[26] M. König and R. Rasch, "Digital teaching an embedded systems course by using simulators," in *2021 ACM/IEEE Workshop on Computer Architecture Education (WCAE)*, 2021.

[27] R. E. Seviora, "A curriculum for embedded system engineering," *ACM Transactions on Embedded Computing Systems*, vol. 4, no. 3, pp. 569–586, 2005.

[28] A. N. Kumar and R. K. Raj, "A first look at the acm/ieee-cs/aaai computer science curricula (cs202x)," in *SIGCSE 2022: Proceedings of the 53rd ACM Technical Symposium on Computer Science Education*, vol. 2, pp. 1023–1024, ACM, 2022.

[29] J. Diz, J. F. Garcia, and J. Dominguez, "Modular architecture with microcontroller for advanced electronic practices," in *2012 Technologies Applied to Electronics Teaching (TAEE)*, pp. 92–97, 2012.

[30] C. Ünsalan, H. D. Gürhan, and M. E. Yücel, *Embedded System Design with ARM Cortex-M Microcontrollers: Applications with C, C++ and MicroPython*. Springer Cham, 1 ed., 2022.

[31] T. Alajbeg and M. Sokele, "Implementation of electronic design automation software tool in the learning process," in *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 532–536, 2019.

[32] F. Karaagac, H. Koc, and S. Ozturk, "An educational electronic design automation tool for high level synthesis and optimization," in *2013 Second International Conference on E-Learning and E-Technologies in Education (ICEEE)*, pp. 165–170, 2013.

[33] N. C. Szekely, S. I. Salcu, C. Popusoi, M. Bojan, and C. Marginean, "Digital electronics for everyone: Custom hands-on experience for students - multiplexing and demultiplexing with the help of an open-hardware platform," in *2023 10th International Conference on Modern Power Systems (MPS)*, pp. 01–04, 2023.

[34] IEEE Spectrum, "Chip hall of fame: Atmel atmega8 the chip at the heart of the original arduino was created by two annoyed students," *History of Technology*, June 30 2017. 2 minute read. https://spectrum.ieee.org/chip-hall-of-fame-atmel-atmega8.

[35] W. Keister, "Logic of relay circuits," *Electrical Engineering*, vol. 68, no. 11, pp. 980–980, 1949.

[36] G. Samara, A. Hussein, I. A. Matarneh, M. Alrefai, and M. Y. Al-Safarini, "Internet of robotic things: Current technologies and applications," in *2021 22nd International Arab Conference on Information Technology (ACIT)*, pp. 1–6, 2021.

[37] J. Krejčí, M. Babiuch, J. Babjak, J. Suder, and R. Wierbica, "Implementation of an embedded system into the internet of robotic things," *Micromachines*, vol. 14, no. 1, 2023.

[38] Z. Gu, Z. Wang, S. Li, and H. Cai, "Design and implementation of an automotive telematics gateway based on virtualization," in *2012 IEEE 15th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops*, pp. 53–58, 2012.

[39] J. Li and Y. Lin, "Iot home automation – smart homes and internet of things," in *2021 3rd International Academic Exchange Conference on Science and Technology Innovation (IAECST)*, pp. 294–298, 2021.

[40] K. P. Nnoli, M. S. Benyeogor, O. O. Olakanmi, and D. A. Umanah, "The computer farmer concept: Human-cyberphysical systems for monitoring and improving agricultural productivity in nigeria," in *2022 IEEE Nigeria 4th International Conference on Disruptive Technologies for Sustainable Development (NIGERCON)*, pp. 1–8, 2022.

[41] W. R. Stevens, *UNIX Network Programming, Volume 1*. Philadelphia, PA: Prentice Hall, 2 ed., Oct. 1997.

[42] S. M. Palakollu, *Socket Programming*, pp. 225–266. Berkeley, CA: Apress, 2021.

[43] M. Xue and C. Zhu, "The socket programming and software design for communication based on client/server," in *2009 Pacific-Asia Conference on Circuits, Communications and Systems*, pp. 775–777, 2009.

[44] A. Saddik, R. Latif, A. E. Ouardi, M. Elhoseny, and A. Khelifi, "Computer development based embedded systems in precision agriculture: tools and application," *Acta Agriculturae Scandinavica, Section B — Soil & Plant Science*, vol. 72, no. 1, pp. 589–611, 2022.

[45] M. Pramanik, M. Khanna, M. Singh, D. Singh, S. Sudhishri, A. Bhatia, and R. Ranjan, "Automation of soil moisture sensor-based basin irrigation system," *Smart Agricultural Technology*, vol. 2, p. 100032, 2022.

[46] K. Obaideen, B. A. Yousef, M. N. AlMallahi, Y. C. Tan, M. Mahmoud, H. Jaber, and M. Ramadan, "An overview of smart irrigation systems using iot," *Energy Nexus*, vol. 7, 2022.

[47] U. Ighrakpata, M. Chouikha, O. A. Ejofodomi, and G. Ofualagba, "Automation of irrigation systems and design of automated irrigation systems," *International Journal Water Resources Management and Irrigation Engineering Research*, vol. 2, pp. 11–27, September 2019.

[48] D. Hu, H. Ke, and W. Fu, "Research and design of control system based on NRF24L01 for intellectualized vehicle," in *2017 6th Data Driven Control and Learning Systems (DDCLS)*, pp. 685–689, 2017.

[49] Xingna, Hou, Jun, Ma, Shouhong, Chen, and Daiyu, Tao, "Design of data collection box based on NRF24L01," *MATEC Web Conf.*, vol. 173, p. 01006, 2018.

[50] Anayatullah, Z. Khan, Z. Muhammad, and A. Saleem, "Web-based distributed control using gprs enabled embedded devices," in *2005 Student Conference on Engineering Sciences and Technology*, pp. 1–6, 2005.

[51] Y.-S. Chen, N. T. Le, M. A. Hossain, A. Islam, D.-y. Kim, Y.-J. Choi, and Y. M. Jang, "Survey of promising technologies for 5g networks," *Mobile Information Systems*, vol. 2016, p. 2676589, 2016.

[52] O. O. Olakanmi, M. S. Benyeogor, K. P. Nnoli, and K. O. Odeyemi, *UAV-Enabled WSN and Communication Framework for Data Security, Acquisition and Monitoring on Large Farms: A Panacea for Real-Time Precision Agriculture*, pp. 17–33. Cham: Springer International Publishing, 2023.

[53] M. S. Benyeogor, O. O. Olakanmi, A. A. Dahiru, S. B. Okoli, K. P. Nnoli, O. I. Lawal, and E. B. Uwak, *A Telematic Control Framework for Multi-actuated Robots using NRF24L01-enabled Multisignal RF Device and Algorithms*. Springer, 2023. In press.

[54] W. Amer, U. Ansari, and A. Ghafoor, "Industrial automation using embedded systems and machine-to-machine, man-to-machine (M2M) connectivity for improved overall equipment effectiveness (OEE)," in *2009 IEEE International Conference on Systems, Man and Cybernetics*, pp. 4450–4454, 2009.

[55] J. Powell, "Profibus and modbus: A comparison." https://www.automation.com/en-us/articles/2013-2/profibus-and-modbus-a-comparison, October 13 2013. Siemens.