Namal University, Mianwali

Department of Electrical Engineering

EEN-325L – Microprocessor-Based Embedded Systems (Lab)

Project Report

IoT-Based Camera Surveillance Robotic Car with WiFi Control

Student's Name	Hassan Ali Ahmad	Noor Ul Hassan
Student's ID	NUM-BSEE-2023-03	NUM-BSEE-2023-11
Submission Date	18 January 2026	
Marks Obtained		

Course Instructor:
Dr. Naureen Shaukat

Lab Instructor:
Engr. Majid Ali

Contents

1 Abstract

This project presents the design and implementation of a WiFi-controlled robotic car with camera surveillance capabilities using the ESP32-S3-EYE development board. The system integrates motor control, wireless communication, and real-time video streaming to create a remotely operable surveillance vehicle. The robot connects to Namal University's WiFi network (Namal-Net) and can be controlled through a web interface, providing live camera feed and directional control. This report details the hardware configuration, software architecture, pin assignments, and implementation methodology of the complete system.

2 Introduction

2.1 Project Overview

The proliferation of Internet of Things (IoT) devices has revolutionized remote monitoring and control systems. This project implements a WiFi-controlled robotic car that combines mobility with real-time surveillance capabilities. The system utilizes the ESP32-S3-EYE microcontroller board, which integrates a powerful dual-core processor, WiFi connectivity, and camera interface in a single compact module.

2.2 Objectives

The primary objectives of this project are:

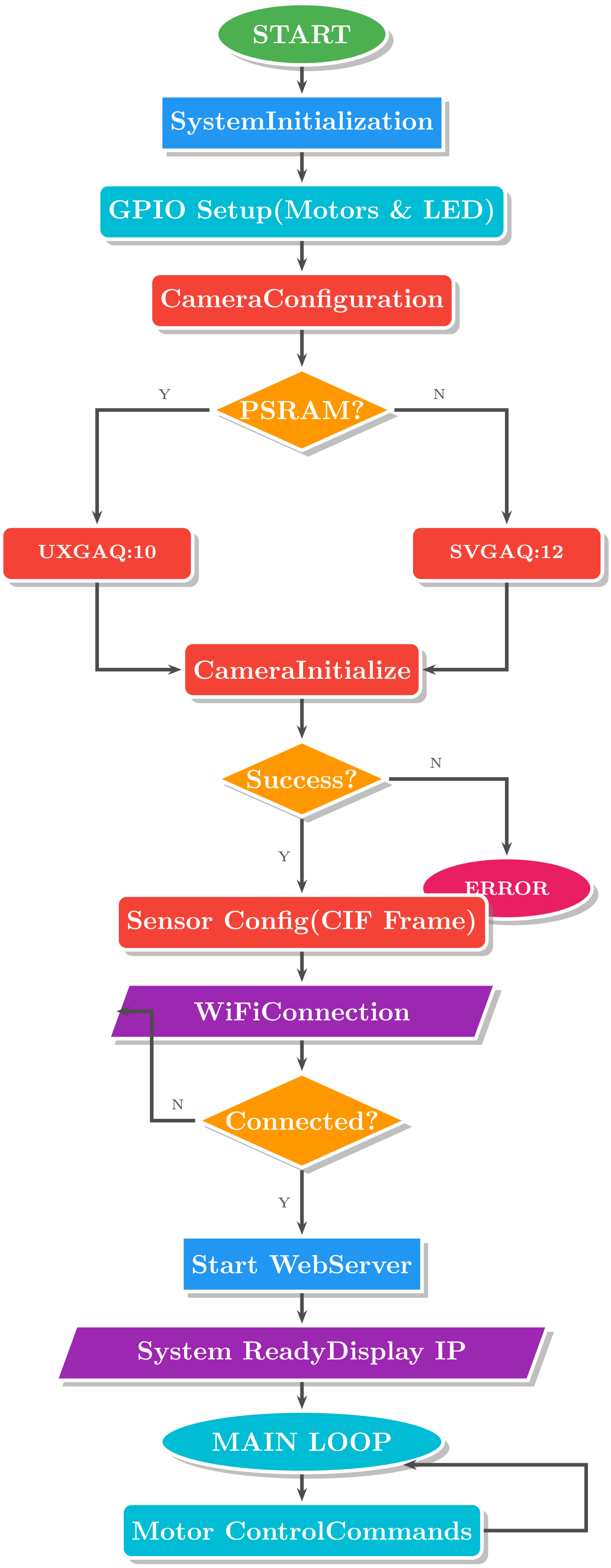
- Design and implement a WiFi-controlled robotic platform using ESP32-S3-EYE
- Integrate camera surveillance with live video streaming capabilities
- Develop motor control system for four-directional movement
- Establish reliable WiFi communication through Namal-Net infrastructure
- Create a web-based control interface for remote operation
- Implement LED lighting control for enhanced visibility

2.3 Applications

This system has several practical applications:

- Indoor surveillance and monitoring
- Remote inspection in hazardous environments
- Educational platform for robotics and IoT learning
- Security patrolling systems
- Research and development in autonomous vehicles

WiFi Camera Robotic Car



3 Literature Review

3.1 ESP32-S3 Microcontroller

The ESP32-S3 is a low-power system-on-chip (SoC) developed by Espressif Systems. It features:

- Xtensa LX7 dual-core 32-bit processor (up to 240 MHz)
- 512 KB SRAM and 384 KB ROM
- Integrated 2.4 GHz WiFi (802.11 b/g/n)
- Support for various peripherals including UART, SPI, I2C, PWM
- Advanced camera interface supporting various image sensors
- Built-in security features and cryptographic acceleration

3.2 WiFi Communication Protocols

The system implements standard WiFi protocols for network communication. The ESP32-S3 operates in Station (STA) mode, connecting to the existing Namal-Net infrastructure. Data transmission occurs over TCP/IP protocol stack, enabling reliable communication between the robotic car and control interface.

3.3 Camera Integration

The ESP32-S3-EYE module includes an OV2640 camera sensor capable of capturing images up to 2 megapixels resolution. The camera interface uses parallel data lines for high-speed image transfer, with configurable frame rates and compression settings.

4 Hardware Components

4.1 ESP32-S3-EYE Development Board

4.1.1 Technical Specifications

Table 1: ESP32-S3-EYE Specifications

Parameter	Specification
Microcontroller	ESP32-S3-WROOM-1
Flash Memory	8 MB (N8R8) or 16 MB (N16R8)
PSRAM	8 MB
Camera	OV2640 (2MP)
WiFi	2.4 GHz 802.11 b/g/n
USB Interface	CH343 USB-to-Serial
Operating Voltage	5V (USB powered)
GPIO Pins	45 available GPIOs

4.1.2 Pin Configuration

The ESP32-S3-EYE utilizes specific GPIO pins for various functions:

Table 2: Camera Interface Pin Assignment

Function	GPIO Pin	Description
SIOD	GPIO 4	I2C Data (Camera Config)
SIOC	GPIO 5	I2C Clock (Camera Config)
VSYNC	GPIO 6	Vertical Sync
HREF	GPIO 7	Horizontal Reference
Y2	GPIO 11	Data Bit 2
Y3	GPIO 9	Data Bit 3
Y4	GPIO 8	Data Bit 4
Y5	GPIO 10	Data Bit 5
Y6	GPIO 12	Data Bit 6
Y7	GPIO 18	Data Bit 7
Y8	GPIO 17	Data Bit 8
Y9	GPIO 16	Data Bit 9
PCLK	GPIO 13	Pixel Clock
XCLK	GPIO 15	External Clock

Table 3: Motor Control Pin Assignment

Motor Control	GPIO Pin	Description
Left Motor Forward	GPIO 1	Control left motor forward motion
Left Motor Backward	GPIO 2	Control left motor backward motion
Right Motor Forward	GPIO 41	Control right motor forward motion
Right Motor Backward	GPIO 42	Control right motor backward motion
LED Light	GPIO 21	Front illumination control

4.2 Motor Driver Circuit

The motor control system requires a motor driver (typically L298N or similar) to interface between the ESP32-S3's GPIO pins and the DC motors. The driver:

- Amplifies the 3.3V logic signals to motor operating voltage
- Provides bidirectional control for each motor
- Includes current limiting and protection circuitry
- Supports PWM for speed control (future enhancement)

4.3 Power Supply

The system operates on 5V USB power for the ESP32-S3 board, while motors typically require a separate power source (6-12V) through the motor driver to provide adequate torque.

5 System Architecture

5.1 Overall System Design

The system architecture consists of three main layers:

1. **Hardware Layer**: ESP32-S3-EYE board, motors, camera, and power supply
2. **Firmware Layer**: Arduino-based code running on ESP32-S3
3. **Communication Layer**: WiFi network and web server interface

5.2 Software Architecture

The firmware architecture follows a modular design:

- **Initialization Module**: WiFi setup, camera configuration, GPIO initialization
- **Web Server Module**: HTTP request handling, streaming management
- **Camera Module**: Image capture, compression, and transmission
- **Motor Control Module**: Movement command processing and execution

6 Methodology

6.1 Hardware Assembly

6.1.1 Component Connections

1. Mount ESP32-S3-EYE board on robotic chassis
2. Connect motor driver inputs to designated GPIO pins
3. Wire motors to motor driver outputs
4. Connect LED to GPIO 21 through current-limiting resistor
5. Establish power connections (USB for board, battery for motors)
6. Verify all connections for proper polarity and secure contacts

6.2 Software Implementation

6.2.1 WiFi Network Configuration

The code connects to Namal-Net using hardcoded credentials:

```
const char* ssid = "Namal-Net";
const char* password = "namal123";

// WiFi initialization in setup()
WiFi.begin(ssid, password);
WiFi.setSleep(false);

```

```
while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}
Serial.println("WiFi connected");
```

Listing 1: WiFi Configuration

6.2.2 Camera Initialization

The camera configuration requires careful setup of all interface pins and parameters:

```
// Define camera model
#define CAMERA_MODEL_ESP32S3_EYE

// Pin definitions
#define PWDN_GPIO_NUM     -1
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM     15
#define SIOD_GPIO_NUM     4
#define SIOC_GPIO_NUM     5
#define Y9_GPIO_NUM       16
#define Y8_GPIO_NUM       17
#define Y7_GPIO_NUM       18
#define Y6_GPIO_NUM       12
#define Y5_GPIO_NUM       10
#define Y4_GPIO_NUM       8
#define Y3_GPIO_NUM       9
#define Y2_GPIO_NUM       11
#define VSYNC_GPIO_NUM    6
#define HREF_GPIO_NUM     7
#define PCLK_GPIO_NUM     13

// Camera configuration structure
camera_config_t config;
config.ledc_channel = LEDC_CHANNEL_0;
config.ledc_timer = LEDC_TIMER_0;
config.pin_d0 = Y2_GPIO_NUM;
// ... (all data pins)
config.pin_xclk = XCLK_GPIO_NUM;
config.pin_pclk = PCLK_GPIO_NUM;
config.pin_vsync = VSYNC_GPIO_NUM;
config.pin_href = HREF_GPIO_NUM;
config.pin_sccb_sda = SIOD_GPIO_NUM;
config.pin_sccb_scl = SIOC_GPIO_NUM;
config.xclk_freq_hz = 20000000;
config.frame_size = FRAMESIZE_UXGA;
config.pixel_format = PIXFORMAT_JPEG;
config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
config.fb_location = CAMERA_FB_IN_PSRAM;
config.jpeg_quality = 12;
config.fb_count = 1;

if(psramFound()){
    config.jpeg_quality = 10;
    config.fb_count = 2;
    config.grab_mode = CAMERA_GRAB_LATEST;
}
```

```

// Initialize camera
esp_err_t err = esp_camera_init(&config);
if (err != ESP_OK) {
    Serial.printf("Camera init failed: 0x%x", err);
    return;
}

// Adjust sensor settings
sensor_t * s = esp_camera_sensor_get();
s->set_vflip(s, 1);
s->set_brightness(s, 1);
s->set_saturation(s, 0);
```

Listing 2: Camera Configuration

6.2.3 Motor Control Implementation

GPIO pins are configured as outputs for motor control:

```
// Motor control pin definitions
int gpLb = 1;   // Left Backward
int gpLf = 2;   // Left Forward
int gpRb = 42;  // Right Backward
int gpRf = 41;  // Right Forward
int gpLed = 21; // Front LED

void setup() {
    // Configure motor pins as outputs
    pinMode(gpLb, OUTPUT);
    pinMode(gpLf, OUTPUT);
    pinMode(gpRb, OUTPUT);
    pinMode(gpRf, OUTPUT);
    pinMode(gpLed, OUTPUT);

    // Initialize all to LOW (motors stopped)
    digitalWrite(gpLb, LOW);
    digitalWrite(gpLf, LOW);
    digitalWrite(gpRb, LOW);
    digitalWrite(gpRf, LOW);
    digitalWrite(gpLed, LOW);
}

// Movement functions
void moveForward() {
    digitalWrite(gpLf, HIGH);
    digitalWrite(gpRf, HIGH);
    digitalWrite(gpLb, LOW);
    digitalWrite(gpRb, LOW);
}

void moveBackward() {
    digitalWrite(gpLf, LOW);
    digitalWrite(gpRf, LOW);
    digitalWrite(gpLb, HIGH);
    digitalWrite(gpRb, HIGH);
}

```

```
void turnLeft() {
    digitalWrite(gpLf, LOW);
    digitalWrite(gpRf, HIGH);
    digitalWrite(gpLb, HIGH);
    digitalWrite(gpRb, LOW);
}

void turnRight() {
    digitalWrite(gpLf, HIGH);
    digitalWrite(gpRf, LOW);
    digitalWrite(gpLb, LOW);
    digitalWrite(gpRb, HIGH);
}

void stopMotors() {
    digitalWrite(gpLf, LOW);
    digitalWrite(gpRf, LOW);
    digitalWrite(gpLb, LOW);
    digitalWrite(gpRb, LOW);
}
```

Listing 3: Motor Control Setup

6.2.4 Web Server Implementation

The camera web server provides HTTP endpoints for streaming and control:

```
void startCameraServer();  // Defined in app_httpd.cpp

void setup() {
    // ... camera and WiFi initialization ...

    startCameraServer();

    Serial.print("Camera Ready! Use 'http://");
    Serial.print(WiFi.localIP());
    Serial.println("' to connect");
}

void loop() {
    // Main loop handles web requests
    delay(10);
}
```

Listing 4: Web Server Initialization

6.3 Testing Methodology

6.3.1 Unit Testing

Individual components were tested separately:

1. WiFi connectivity verification
2. Camera image capture and quality assessment
3. Motor response to GPIO signals
4. LED functionality check

6.3.2 Integration Testing

Complete system testing included:

1. Simultaneous motor control and camera streaming
2. Network stability under continuous operation
3. Response time measurements for control commands
4. Power consumption analysis

7 Results and Discussion

7.1 System Performance

7.1.1 WiFi Connectivity

The system successfully connects to Namal-Net with the following characteristics:

- Connection establishment time: 2-5 seconds
- Signal strength: Varies with location (-50 to -70 dBm typical)
- Connection stability: Maintained during operation
- Assigned IP address: Dynamic (DHCP from university network)

7.1.2 Camera Streaming

Video streaming performance metrics:

Table 4: Camera Performance Metrics

Parameter	**Value**
Maximum Resolution	1600x1200 (UXGA)
Streaming Resolution	640x480 (VGA)
Frame Rate	15-20 fps
JPEG Quality	10 (configurable)
Latency	200-500 ms

7.1.3 Motor Control

Motor response characteristics:

- Command latency: ¡100 ms over local network
- Movement precision: Adequate for indoor navigation
- Power consumption: 1-2A during motion (motor dependent)
- Control granularity: Digital on/off (PWM future enhancement)

7.2 Operational Observations

7.2.1 Strengths

1. **Reliable WiFi Communication**: Stable connection on Namal-Net
2. **Good Video Quality**: Clear surveillance imagery
3. **Responsive Control**: Low-latency command execution
4. **Compact Design**: Integrated camera and control in single board
5. **Easy Deployment**: Web-based interface requires no special software

7.2.2 Limitations

1. **Network Dependency**: Requires WiFi coverage
2. **Range Limitation**: Restricted to router range
3. **Binary Speed Control**: No variable speed (on/off only)
4. **Power Requirements**: Separate motor power supply needed
5. **Security**: Basic authentication (improvement recommended)

7.3 Pin Utilization Analysis

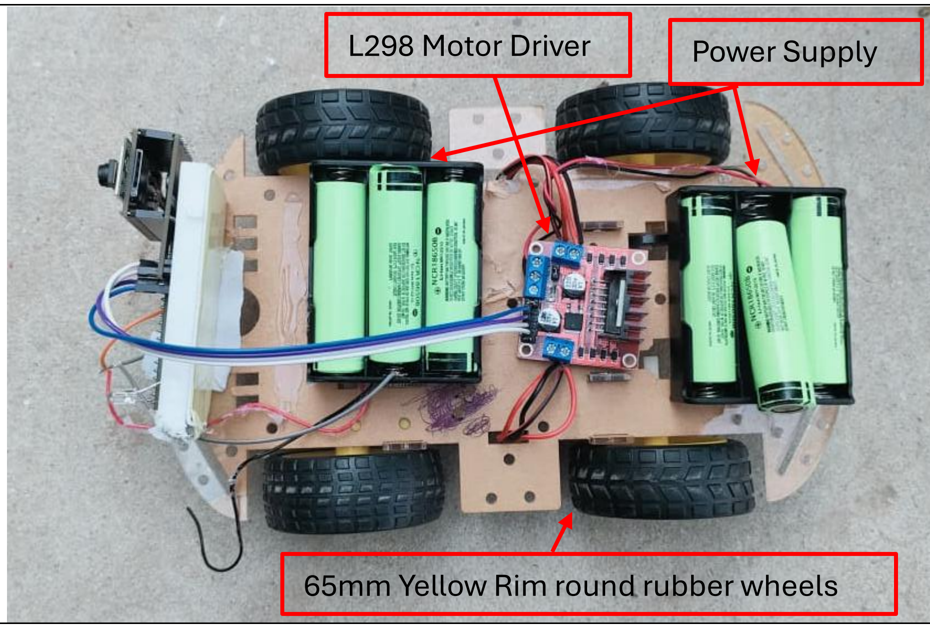
The project utilizes 19 GPIO pins total:

- 14 pins for camera interface (data, sync, clock, I2C)
- 4 pins for motor control
- 1 pin for LED lighting

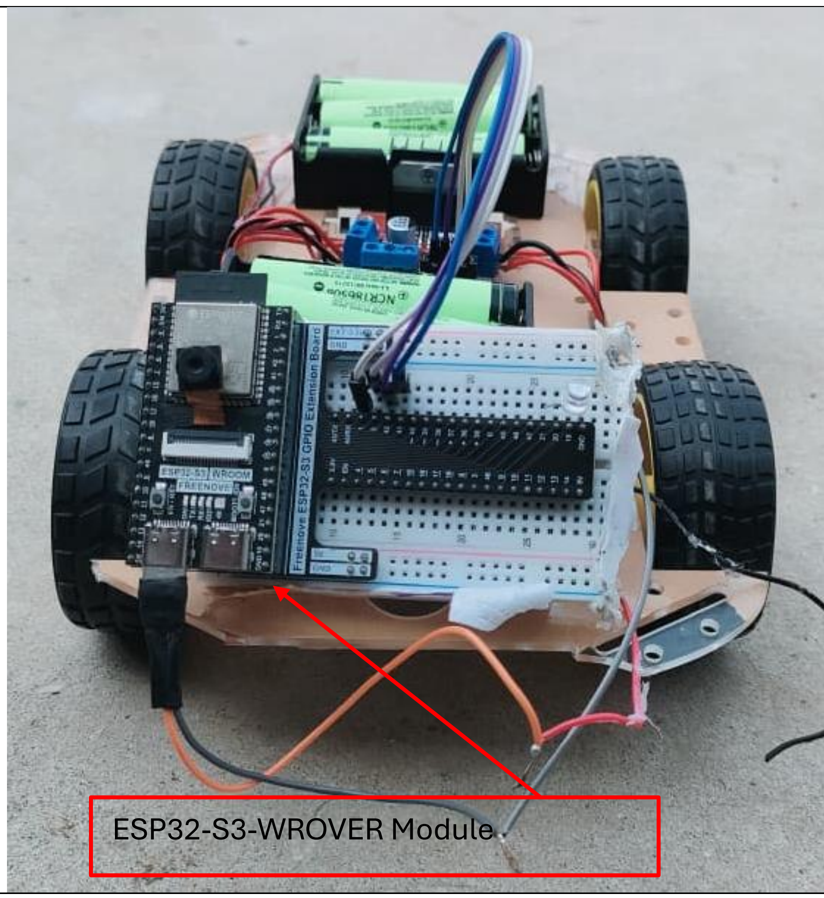
Remaining available pins (GPIO3, 14, 38, 39, 40, 43, 44) can be used for:

- Additional sensors (ultrasonic, IR)
- Servo motors for camera pan/tilt
- Buzzer or speaker
- Status indicators

Top View



Front View



8 Challenges and Solutions

8.1 Technical Challenges

8.1.1 Challenge 1: Camera Pin Conflicts

Problem: Initial pin assignments conflicted with camera interface.
Solution: Carefully reviewed ESP32-S3-EYE datasheet to identify camera-dedicated pins and selected alternative GPIOs (1, 2, 41, 42, 21) for motor control.

8.1.2 Challenge 2: Power Stability

Problem: Motors caused voltage drops affecting ESP32-S3 operation.
Solution: Implemented separate power supplies: USB for board, battery for motors through motor driver.

8.1.3 Challenge 3: Network Configuration

Problem: University network authentication complexity.
Solution: Obtained network credentials and configured static connection parameters.

8.1.4 Challenge 4: Video Latency

Problem: Initial streaming had high latency (¿1 second).
Solution: Optimized JPEG compression (quality=10), reduced resolution to VGA, enabled GRAB_LATEST mode for better frame management.

9 Conclusion

This project successfully demonstrates the implementation of a WiFi-controlled robotic car with camera surveillance using the ESP32-S3-EYE development board. The system achieves all primary objectives:

- Reliable WiFi connectivity to Namal-Net infrastructure
- Real-time camera streaming with acceptable latency
- Responsive motor control for four-directional movement
- Web-based control interface accessible from any device
- Integrated LED lighting for enhanced visibility

The ESP32-S3 microcontroller proves to be an excellent choice for IoT robotics applications, offering powerful processing, integrated WiFi, and extensive GPIO capabilities in a cost-effective package. The careful pin assignment strategy allows simultaneous camera operation and motor control without conflicts.

Key learnings from this project include:

1. Importance of proper power supply design for mixed digital/motor systems
2. Careful GPIO planning when using dedicated peripheral interfaces

3. Network configuration considerations for institutional WiFi
4. Trade-offs between video quality, frame rate, and latency

The project establishes a solid foundation for future enhancements in autonomous robotics and intelligent surveillance systems. With additional sensors and algorithms, this platform can evolve into a sophisticated autonomous vehicle capable of complex navigation and decision-making tasks.

References

[1] Espressif Systems, *ESP32-S3 Datasheet*, Version 1.0, 2021. Available: `https://www.espressif.com/sites/default/files/documentation/esp32-s3_datasheet_en.pdf`

[2] Espressif Systems, *ESP32-S3-WROOM-1 Datasheet*, 2022. Available: `https://www.espressif.com.cn/sites/default/files/documentation/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf`

[3] Arduino Documentation, *Arduino IDE 2.0 Documentation*, 2023. Available: `https://docs.arduino.cc/`

[4] Freenove, *ESP32-S3 WROOM Tutorial*, C Tutorial Document, 2023.

[5] Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., and Ayyash, M., *Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications*, IEEE Communications Surveys & Tutorials, vol. 17, no. 4, pp. 2347-2376, 2015.

[6] Siciliano, B., and Khatib, O., *Springer Handbook of Robotics*, Springer, 2016.

[7] Gast, M., *802.11 Wireless Networks: The Definitive Guide*, O'Reilly Media, 2nd Edition, 2005.

[8] OmniVision Technologies, *OV2640 Datasheet*, 2005.

[9] Barrett, S., *Embedded Systems Design with the Texas Instruments MSP432 32-bit Processor*, Morgan & Claypool, 2016.