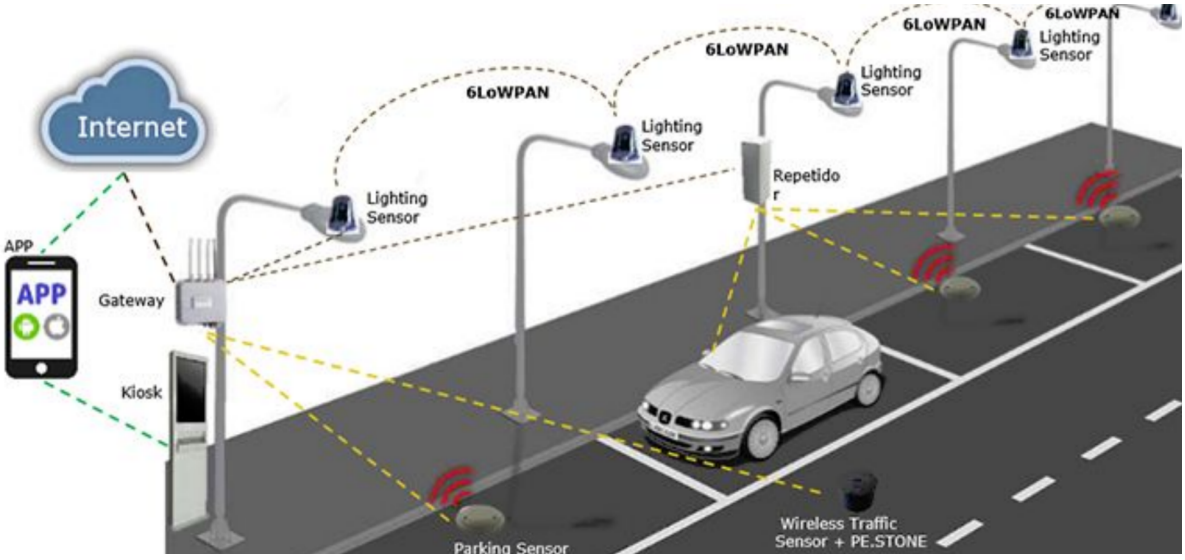


CIE 408 - Embedded Systems

Final Project - Smart Parking System

Spring 2023

Team Members:

Youssef Islam Elshabrawy 201-900-667

Amr Ahmed Elmasry 201-901-202

Ahmed Mohamed Ibrahem 201-902-227

مدينة زويل

ZEWAIL CITY

ESTABLISHED 2000

INAUGURATED 2011

Table Of Contents

1. Abstract

This report presents a comprehensive parking system that utilizes Internet of Things (IoT) capabilities and advanced technologies to address the challenges posed by population growth and urban mobility. The system includes a mobile application that provides real-time data on available parking spaces, incorporating factors such as parking rates and amenities. Reservation capabilities allow drivers to secure parking spaces while on the move. Hardware components such as Arduino, ESP module, and ultrasonic sensors are utilized to facilitate data collection and communication. Future work includes expanding the system to various locations, integrating GPS technology, enhancing user experience, and incorporating features like automatic time-based parking calculators, automatic visa payments, and electric car charging. This innovative parking system aims to optimize parking availability, streamline the parking experience, and promote sustainable urban development.

2. Introduction

2.1 The Problem

The exponential growth of the global population has brought about a host of challenges and complications, particularly in the domain of transportation. With more people residing in cities and an increasing number of vehicles on the roads, several pressing issues have emerged, including heightened pollution levels and aggravating traffic congestion. Unfortunately, the existing transportation infrastructure often fails to keep pace with the expanding population, resulting in a significant strain on urban mobility systems.

Furthermore, the lack of effective integration of smart city solutions, such as innovative parking arrangements, has only exacerbated the problem. As a result, finding suitable parking spaces in congested areas has become a time-consuming and frustrating endeavor for drivers. The absence of real-time information on available parking spots and the inability to efficiently navigate through parking lots have contributed to both traffic congestion and driver inconvenience.

2.1 Solution

Amidst these challenges, a viable solution emerges: envision a future where vehicles seamlessly communicate with the city's traffic agency and leverage the

power of advanced technologies. By incorporating Internet of Things (IoT) capabilities, a transformative solution can be implemented to address the issues posed by population growth and enhance urban mobility.

One such solution involves the development of a comprehensive parking application that revolutionizes the way drivers approach parking. This innovative app would provide real-time data on nearby available parking spaces, enabling users to make informed decisions based on their preferences. The application would offer a convenient scheme, suggesting the most optimal parking approach tailored to the driver's requirements.

In addition to displaying available parking options, the app would also take into account factors such as parking rates, distance from the desired destination, and availability of additional amenities. This empowers drivers to choose between the most cost-effective or closest parking options, ensuring a seamless and hassle-free parking experience.

Furthermore, the integration of reservation capabilities within the app allows drivers to secure a parking spot while on the move. Whether for important appointments or last-minute plans, this feature ensures that a designated parking space is available upon arrival, saving time and reducing stress.

Leading companies, such as Volkswagen (VW) and T-Mobile, have recognized the transformative potential of IoT solutions for smart cities and are actively investing in this domain. These forward-thinking organizations are not only funding research and development but also conducting pilot projects worldwide to demonstrate the effectiveness and feasibility of these smart mobility solutions. By embracing advanced technologies and fostering collaboration between public and private sectors, they are at the forefront of shaping a future where smart mobility seamlessly integrates with our daily lives, promoting efficient transportation and sustainable urban development.

3. Hardware Description

3.1 Project

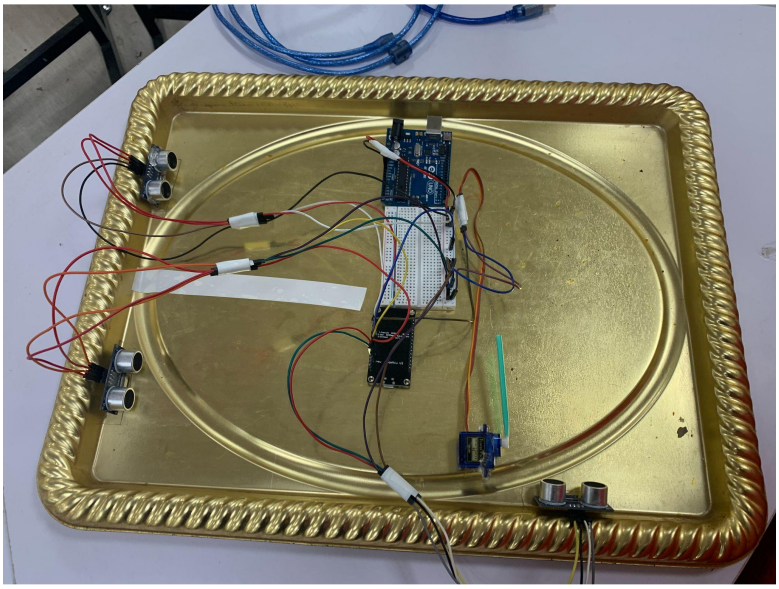


Figure: Project Prototype

In our proposed system, we aim to create a simulated entrance parking gate with two available parking slots. The system operates as follows:

1. **Car Detection and Counting:** We implement a mechanism to detect and count the cars already present in the parking lot.
2. **Information Collection and Transmission:** The collected car count information is then transmitted to either a screen located in front of the parking lot or to a cloud server. This information serves as the basis for informing the driver about the current state of the parking lot.
3. **Driver Notification:** Using a phone application, the server accesses the transmitted information and notifies the driver accordingly. If the parking lot is full, the gate remains closed and the driver is informed about the unavailability of slots. Conversely, if there are available slots, the gate opens, allowing the driver to enter. Additionally, the system keeps track of the number of cars that enter the parking lot.
4. **Slot Mapping**: The application provides a map displaying the available slots along with their corresponding numbers and map locations. This feature assists the driver in locating an appropriate parking spot.
5. **Parking Status Update:** When a car parks in a specific slot, the corresponding slot indicator turns red, indicating its occupation.

By presenting the information in a more structured and concise manner, the revised description provides a clearer understanding of the system's functionality.

3.2 Block Diagram

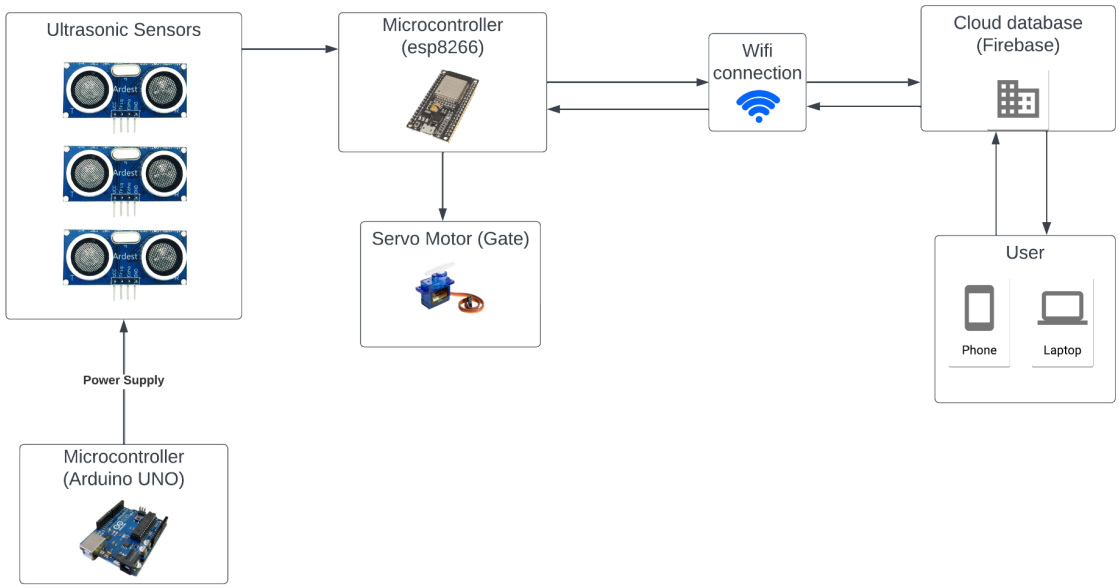


Figure: Functional Block Diagram

In the block diagram, the Arduino functions as a power supply for various sensors, while the ESP module serves as the main microcontroller due to its cost-effectiveness. The ESP module incorporates a WiFi module, enabling it to establish a wireless connection for transmitting and receiving data. This data is transmitted to the Firebase Real-Time Database, leveraging its capabilities for efficient data storage and retrieval. The mobile application, used by drivers, interacts with the Firebase Database, providing drivers with access to real-time information regarding parking slot availability and other related details.

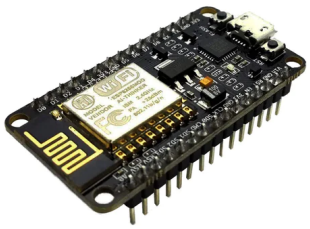
3.3 Components

3.3.1 Arduino UNO

The Arduino board serves as a reliable power source, providing a steady supply of 5 volts to the system.

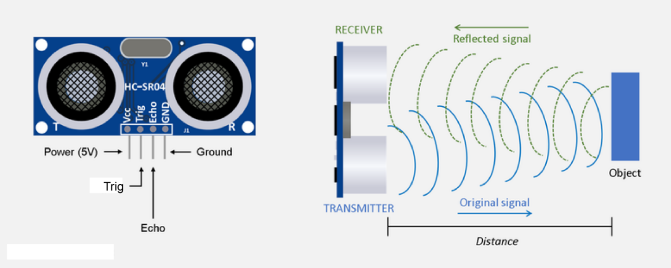
3.3.2 ESP8266

The ESP module takes on the crucial role of sensor reading and communication. It gathers data from various sensors and seamlessly communicates with the Firebase Realtime Database, ensuring a constant flow of information to the application.



3.3.3 Ultrasonic Sensors

Ultrasonic sensors play a pivotal role in the system, enabling the detection of parking slot occupancy. Additionally, these sensors are responsible for managing the opening and closing of the parking entrance gate, ensuring smooth and efficient access control.



3.3.4 Servo Motor

A servo motor is employed to seamlessly operate the opening and closing mechanism of the parking entrance gate. Its precise control and smooth movement ensure efficient and reliable gate operation.

3.3.5 App Inventor Application

We utilize the power of MIT App Inventor to craft our application, which seamlessly interacts with the real-time Firebase database. This user-friendly platform empowers us to build an intuitive and engaging app that leverages the data from the Firebase database in real-time.

3.4 Design Description

The main used component is the ESP8266, as it controls all the logic and acts as the brain of the hardware implementation. It sends and receives signals from the other components and acts based on the written software on its memory. The ESP8266 is connected to sensors and actuators, sensors such as the ultrasonic sensor, and actuators such as the servo motor. ESP8266 utilizes the other

components to reach the functionality of the project. Arduino is used as a power supply to power the sensors and the actuators.

4. Software Design

4.1 Flow Chart

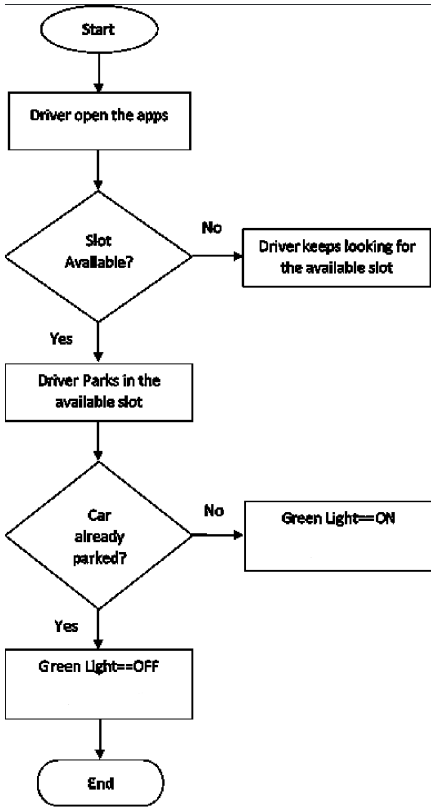


Figure: Logic Flow Chart

As we observe the flowchart, we can depict the following sequence of events:

1. The driver initiates the app.
2. The app determines the availability of a parking slot.
    a. If no slots are available:
        i. The app displays a message indicating the unavailability of slots in the parking.
        ii. The gate remains closed.
    b. If slots are available:
        i. The gate opens.
3. We proceed to the designated parking slot
    a. If the slot is occupied by a car:
        i. The slot indicator turns red.

b. If the slot is unoccupied:
    i. The slot indicator turns green.

In this process, the driver starts by launching the app. Subsequently, the app checks if any parking slots are vacant. If no slots are available, the app notifies the driver about the unavailability, resulting in the gate remaining closed. However, if slots are found to be vacant, the gate opens, granting access to the parking area. After entering the parking area, we reach the specified slot. If a car is already occupying the slot, the slot indicator turns red to indicate its unavailability. Conversely, if the slot is unoccupied, the indicator turns green, signaling that the driver can proceed to park the car.

4.2 Main Code Components

4.2.1 Initialization

```
#include <ESP8266WiFi.h>
#include <Firebase_ESP_Client.h>
#include <Servo.h>
#include                "addons/RTDBHelper.h"
#include                "addons/TokenHelper.h"
#define   API_KEY       "AIzaSyA_aZ0aGPDKSUdj06_aU-3ENg1PtH19K5M"
#define   DATABASE_URL  "https://cie408---project-default-rtdb.firebaseio.com/"
#define   ssid          "Youssef"
#define   pass          "1234556789"
FirebaseData            fbdo;
FirebaseAuth            auth;
FirebaseConfig          config;
bool   signupok        = false;
Servo motor; //servo motor object
//Two car slots and gate sensor (3 ultrasonic sensors)
#define echo1 D6
#define trig1 D5
#define echo2 D1
#define trig2 D0
#define echo3 D8
#define trig3 D7
int echos[] = {D6, D1}; //List of echo signals of car slots sensors
int trigs[] = {D5, D0}; //List of trigger signals of car slots sensors
int cars = 0; //number of cars inside the parking
int cars_max = 2; //maximum number of cars allowed in the parking
```

In the initialization step, we include necessary libraries and set the variables and the parameters to be used in the program. We initialize the sensors' pins on the

ESP8266, we also create necessary data structures, as well as defining the parameters needed to connect the ESP8266 to the WIFI and the Firebase.

4.2.2 Setup function

```
void setup() {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid,pass);
  while(WiFi.status() != WL_CONNECTED){
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to:");
  Serial.print(ssid);
  Serial.println("");
  Serial.print("IP Address: ");
  Serial.print(WiFi.localIP());
  config.api_key = API_KEY;
  config.database_url= DATABASE_URL;
  if(Firebase.signUp(&config,&auth,"","")){
    Serial.println("OK");
    signupok=true;
  }
  else{
    Serial.println("Error");
  }
  Firebase.begin(&config,&auth);
  Firebase.reconnectWiFi(true);

  motor.attach(D2);
  motor.write(0);
  pinMode(trig1, OUTPUT);
  pinMode(trig2, OUTPUT);
  pinMode(trig3, OUTPUT);
  pinMode(echo1, INPUT);
  pinMode(echo2, INPUT);
  pinMode(echo3, INPUT);
}
```

In this part, we set up the code that runs only once at the beginning of the program. Here we connect the ESP8266 to the WIFI and also to the Firebase.we also set up the servo motor pin, the sensors' pin modes, and the serial.

4.2.3 Loop function

```
void loop() {
  Firebase.RTDB.setString(&fbdo, "/Cars", cars);
  for(int i = 0; i < cars_max; i++){
    int trig = trigs[i];
    int echo = echos[i];
    digitalWrite(trig, LOW);
    delayMicroseconds(2);
    digitalWrite(trig, HIGH);
    delayMicroseconds(10);
    digitalWrite(trig, LOW);
    long time = pulseIn(echo, HIGH);
    double dis = time * 0.034 / 2;//in cm
    if (dis <= 10){
      if (state[i] == "Empty"){
        Firebase.RTDB.setString(&fbdo, s[i], "Full");
        state[i] = "Full";
      }
    }
    else{
      if (state[i] == "Full"){
        Firebase.RTDB.setString(&fbdo, s[i], "Empty");
        state[i] = "Empty";
      }
    }
  }

  if(cars != cars_max){
    //Entrance
    digitalWrite(trig3, LOW);
    delayMicroseconds(2);
    digitalWrite(trig3, HIGH);
    delayMicroseconds(10);
    digitalWrite(trig3, LOW);
    long time3 = pulseIn(echo3, HIGH);
    double dis3 = time3 * 0.034 / 2;//in cm
    if (dis3 <= 10){
      motor.write(180);
      Firebase.RTDB.setString(&fbdo, "Gate", "Open");
      delay(3000);
      motor.write(0);
      Firebase.RTDB.setString(&fbdo, "Gate", "Close");
      cars++;
    }
  }
}
```

In this part, we write all the logic that runs the program continuously. We loop through each car slot sensor to check whether the slot is empty or not and write the data onto the Firebase. We also check the gate sensor to examine whether there is a car willing to get inside the parking or not, and if there exists a car and there exists empty space inside the parking, we open the gate for it and

increment the number of cars inside the parking by one then update the Firebase by the number of cars..

5. Testing Results

5.1 Screenshot of the Firebase

5.2 Video link of the prototype running

Link :CIE 408 - Project video

6. Conclusion

In conclusion, we have successfully addressed the smart mobility problem by proposing a scalable solution. Our small-scale system effectively tackles the challenges at hand, providing a foundation for future expansion and growth.

6.1 Future work

The future development of the system can include the following aspects:

- **Expansion and Deployment:** Expanding the system's implementation to various parking locations ensures its widespread availability and usability, benefiting a larger user base.

- **Establishment of Dedicated Traffic Agencies:** Creating dedicated traffic agencies for each local area can enhance management and coordination. These agencies can oversee parking regulations, monitor traffic flow, and implement strategies to optimize parking availability and efficiency.

- **Integration of GPS Technology:** Integrating GPS technology into the mobile application enables seamless car detection and real-time

connection to the nearest available parking spaces. This feature enhances convenience and reduces the time spent searching for parking.

- **Enhancement of User Experience:** Improving user experience can be achieved through additional features like SMS notifications. By providing advance information about the unavailability of nearby parking slots before entering the parking area, drivers can save time and avoid unnecessary congestion.

Additionally, to further enhance the parking system and provide an optimal parking experience, the following additions and improvements can be considered:

1. **Automatic Time-Based Parking Calculator:** Implementing an automatic time-based parking calculator within the mobile application can save drivers the hassle of manually calculating their parking duration. By integrating with the system's data, the application can accurately determine the parking time and calculate the corresponding fees.

2. **Automatic Visa Payments:** Incorporating a secure payment gateway into the mobile application enables drivers to make automatic visa payments for parking fees. This feature eliminates the need for physical transactions, streamlining the payment process and saving time for both drivers and parking management.

3. **Electric Car Charging Integration:** In anticipation of the growing popularity of electric vehicles, future iterations of the system can include electric car charging stations. By integrating charging infrastructure, the parking system caters to the needs of electric vehicle owners, providing a comprehensive parking and charging solution.

By incorporating these future developments, the parking system can evolve into a comprehensive solution that not only optimizes parking availability but also enhances user convenience and supports sustainable transportation options.

Appendix A

A.1 Ultrasonic sensor

Ultrasonic Ranging Module HC - SR04

Product features:

Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of work:
(1) Using IO trigger for at least 10us high level signal,
(2) The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.
(3) IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic to returning.
Test distance = (high level time×velocity of sound (340M/S) / 2,

Wire connecting direct as following:

- 5V Supply
- Trigger Pulse Input
- Echo Pulse Output
- 0V Ground

Electric Parameter

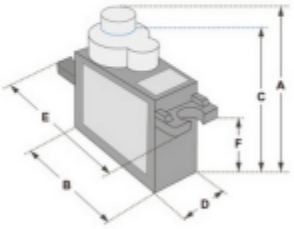
Working Voltage	DC 5 V
Working Current	15mA
Working Frequency	40Hz
Max Range	4m
Min Range	2cm
MeasuringAngle	15 degree
Trigger Input Signal	10uS TTL pulse
Echo Output Signal	Input TTL lever signal and the range in proportion
Dimension	45*20*15mm

Datasheet Link: Ultrasonic HC-SR04

A.2 Servo motor

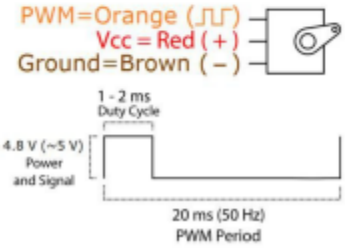
SERVO MOTOR SG90 DATA SHEET

Tiny and lightweight with high output power. Servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but smaller. You can use any servo code, hardware or library to control these servos. Good for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. It comes with a 3 horns (arms) and hardware.



Dimensions & Specifications
A (mm) : 32
B (mm) : 23
C (mm) : 28.5
D (mm) : 12
E (mm) : 32
F (mm) : 19.5
Speed (sec) : 0.1
Torque (kg-cm) : 2.5
Weight (g) : 14.7
Voltage : 4.8 - 6

Position "0" (1.5 ms pulse) is middle, "90" (~2ms pulse) is middle, is all the way to the right, "-90" (~1ms pulse) is all the way to the left.



Datasheet Link: [Servo Motor SG90]

A.3 Arduino Uno

Features

- **ATMega328P** Processor
  - **Memory**
    - AVR CPU at up to 16 MHz
    - 32KB Flash
    - 2KB SRAM
    - 1KB EEPROM
  - **Security**
    - Power On Reset (POR)
    - Brown Out Detection (BOD)
  - **Peripherals**
    - 2x 8-bit Timer/Counter with a dedicated period register and compare channels
    - 1x 16-bit Timer/Counter with a dedicated period register, input capture and compare channels
    - 1x USART with fractional baud rate generator and start-of-frame detection
    - 1x controller/peripheral Serial Peripheral Interface (SPI)
    - 1x Dual mode controller/peripheral I2C
    - 1x Analog Comparator (AC) with a scalable reference input
    - Watchdog Timer with separate on-chip oscillator
    - Six PWM channels
    - Interrupt and wake-up on pin change
  - **ATMega16U2 Processor**
    - 8-bit AVR® RISC-based microcontroller
  - **Memory**
    - 16 KB ISP Flash
    - 512B EEPROM
    - 512B SRAM
    - debugWIRE interface for on-chip debugging and programming
  - **Power**
    - 2.7-5.5 volts

Datasheet Link: [Arduino Uno A000066]

A.4 ESP8266

Specifications

Table 1-1. Specifications

Categories	Items	Parameters
Wi-Fi	Certification	Wi-Fi Alliance
	Protocols	802.11 b/g/n (HT20)
	Frequency Range	2.4 GHz ~ 2.5 GHz (2400 MHz ~ 2483.5 MHz)
	TX Power	802.11 b: +20 dBm
		802.11 g: +17 dBm
		802.11 n: +14 dBm
	Rx Sensitivity	802.11 b: –91 dbm (11 Mbps)
		802.11 g: –75 dbm (54 Mbps)
		802.11 n: –72 dbm (MCS7)
	Antenna	PCB Trace, External, IPEX Connector, Ceramic Chip
Hardware	CPU	Tensilica L106 32-bit processor
	Peripheral Interface	UART/SDIO/SPI/I2C/I2S/IR Remote Control
		GPIO/ADC/PWM/LED Light & Button
	Operating Voltage	2.5 V ~ 3.6 V
	Operating Current	Average value: 80 mA
	Operating Temperature Range	–40 °C ~ 125 °C
	Package Size	QFN32-pin (5 mm x 5 mm)
	External Interface	-
Software	Wi-Fi Mode	Station/SoftAP/SoftAP+Station
	Security	WPA/WPA2
	Encryption	WEP/TKIP/AES
	Firmware Upgrade	UART Download / OTA (via network)
	Software Development	Supports Cloud Server Development / Firmware and SDK for fast on-chip programming
	Network Protocols	IPv4, TCP/UDP/HTTP
	User Configuration	AT Instruction Set, Cloud Server, Android/iOS App

Datasheet Link: ESP8266