Arduino to MongoDB Data Integration Using Node.js

This documentation outlines the procedure to set up a system where data from an Arduino device is read through a serial port using Node.js and stored in a MongoDB database. This guide is intended for students and developers looking to implement real-time data processing and storage solutions.

Prerequisites

Hardware:

- Arduino board

Software:

- Arduino IDE
- Node.js
- MongoDB (Local or MongoDB Atlas)
- A code editor (e.g., Visual Studio Code)

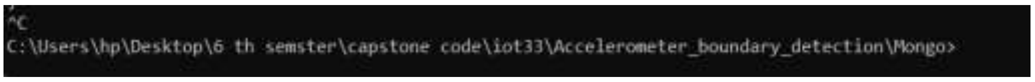
1. Setting Up the Environment

1.1 Installing Necessary Software

- **Node.js:** Download and install Node.js from nodejs.org.
- **Arduino IDE:** Download from Arduino.cc and install it based on your operating system.

1.2 Project Initialization

- Create a directory for your project on your computer using mkdir command.



```
^C
C:\Users\hp\Desktop\6 th semster\capstone code\iot33\Accelerometer_boundary_detection\Mongo>
```

- Open a terminal or command prompt, navigate to the project directory, and run:

```bash
npm init -y
```

- Install required Node.js packages

```bash
npm install serialport mongodb mongoose
```

2. Arduino Configuration

2.1 Arduino Sketch

Upload a simple sketch to your Arduino to send sensor data through the serial port. Here’s an example sketch that sends analogue sensor readings:

```cpp
void setup() {
  Serial.begin(9600);
}

void loop() {
  int sensorValue = analogRead(A0);
  Serial.println(sensorValue);
  delay(1000);
}
```

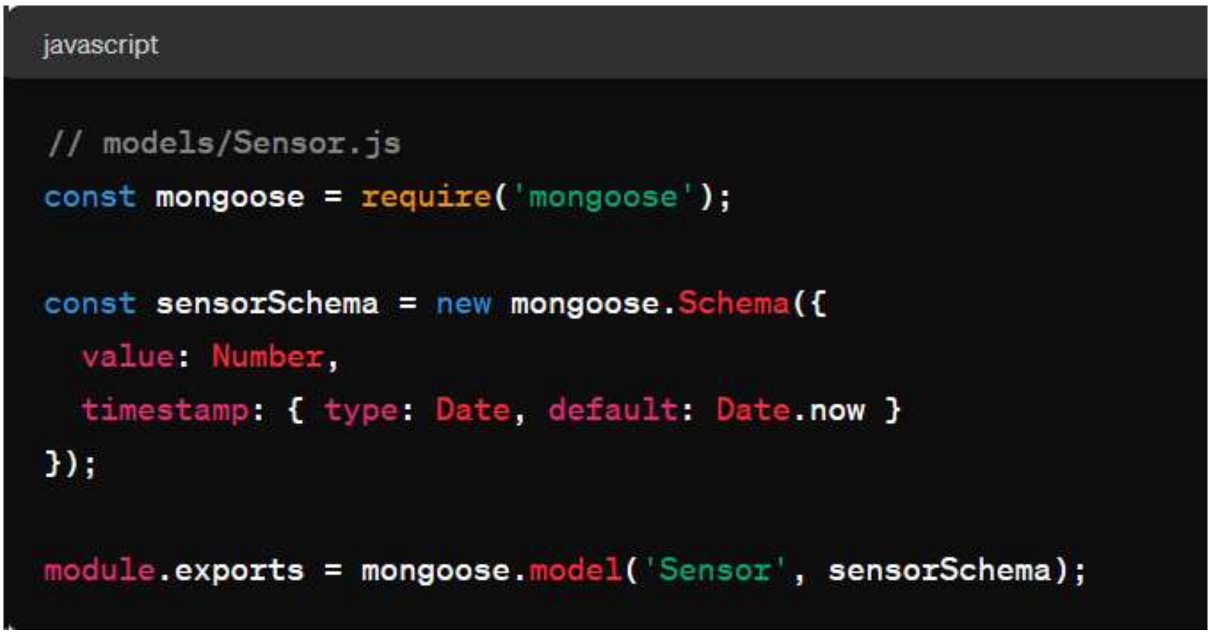
- Connect your Arduino to your PC.
- Open the Arduino IDE, select the right board and port under Tools menu.
- Paste the above code and upload it to the Arduino.

Note: In my case, I am using Arduino IOT33 inbuilt accelerometer sensor LSM3D6, you can upload the code of necessity in Arduino IDE method of uploading the code to the online server will be the same.

3. Node.js Application Development

3.1 Creating Sensor Data Model

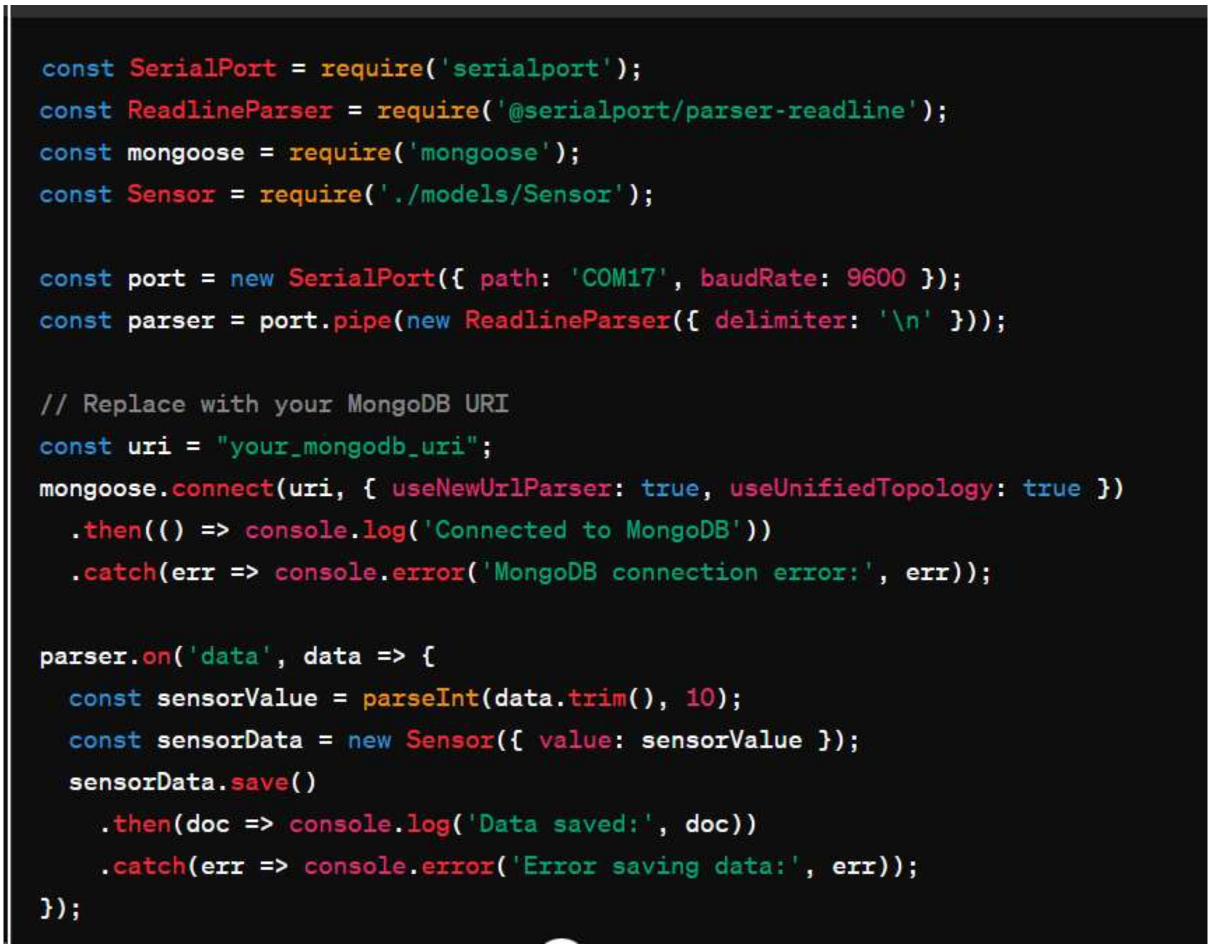
- Create a Sensor.js file in your project to define the MongoDB data schema using Mongoose:



```javascript
// models/Sensor.js
const mongoose = require('mongoose');

const sensorSchema = new mongoose.Schema({
  value: Number,
  timestamp: { type: Date, default: Date.now }
});

module.exports = mongoose.model('Sensor', sensorSchema);
```

3.2 Writing the Node.js Script

- Create an index.js file to handle serial communication and database operations:

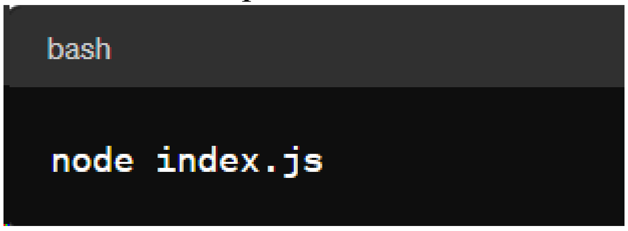


```
const SerialPort = require('serialport');
const ReadlineParser = require('@serialport/parser-readline');
const mongoose = require('mongoose');
const Sensor = require('./models/Sensor');

const port = new SerialPort({ path: 'COM17', baudRate: 9600 });
const parser = port.pipe(new ReadlineParser({ delimiter: '\n' }));

// Replace with your MongoDB URI
const uri = "your_mongodb_uri";
mongoose.connect(uri, { useNewUrlParser: true, useUnifiedTopology: true })
  .then(() => console.log('Connected to MongoDB'))
  .catch(err => console.error('MongoDB connection error:', err));

parser.on('data', data => {
  const sensorValue = parseInt(data.trim(), 10);
  const sensorData = new Sensor({ value: sensorValue });
  sensorData.save()
    .then(doc => console.log('Data saved:', doc))
    .catch(err => console.error('Error saving data:', err));
});
```

- Replace 'COM17' with the correct COM port for your Arduino.
- Replace "your_mongodb_uri" with your actual MongoDB connection string.
- To know how to get the connection string visit → https://docs.google.com/document/d/1ayWxjZcDFhHgOdKM1usIA538ZhKWjqyi/edit?usp=sharing&ouid=113490668225413126773&rtpof=true&sd=true

3.3 Running the Script

- Open a terminal.
- Navigate to your project directory.
- Execute the script



bash

```bash
node index.js
```

4. Troubleshooting and Common Issues

4.1 Serial Port Not Found

- Ensure the Arduino is properly connected and recognized by your computer.
- Check the correct COM port in the Device Manager (Windows) or System Report (macOS).

4.2 MongoDB Connection Issues

- Verify the MongoDB URI.
- Ensure network access is allowed from your IP to MongoDB Atlas if using cloud.

5. Conclusion

This guide provides a detailed approach to setting up a system that reads data from an Arduino and stores it in MongoDB via Node.js. It is designed to help students and developers create a real-time data logging system efficiently.