**Department: MCA Academic Year: 2023-24**

**Year: FY MCA Seminar/Project: PBL Project**

|  |
| --- |
| **Group No.: 5** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Seat No.** | **Student Name** | | **Sign.** | **Guide Name** | **Guide Sign.** |
| 3505 | Bandagar Malashri | |  | Dr.Atul Newase |  |
| 3534 | Gawade Shivam | |  |
| 3538 | Habib Prathamesh | |  |
| **Project Sponsored: Y/N** | | **Project Sponsored By: NA** | | | |

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## Declaration of Novelty and Originality

**Novelty Based on Current Knowledge**

During our collaboration on this "Accurate Clock using Arduino" project, we have focused on achieving high accuracy. To the best of my knowledge and based on my access to a vast dataset of information, the following aspects of our design could potentially contribute to the novelty of the invention:

**Unique Hardware Combination:** (Specify here if your design uses components beyond the basic Arduino board in a novel way). For instance, if you're using a specific sensor or component combination not typically used for high-accuracy clocks with Arduino, you can mention it here.

**Novel Programming Approach:** (Describe here if your code incorporates a unique algorithm or method to achieve high accuracy). This could involve innovative calibration techniques, temperature compensation methods, or other programming aspects that distinguish your approach from standard Arduino clock programs.

**External Time Synchronization Method:** (Explain here if your clock synchronizes with external time sources in a new way). This could involve using a novel communication protocol, a unique data format, or a custom server interaction method for time updates.

**Disclaimer and Recommendation**

It's important to remember that this is a preliminary assessment based on our current work. A thorough novelty search conducted by a patent attorney is crucial to definitively determine the patentability of your invention. They can investigate existing patents and publications to assess the originality of your design.

**Moving Forward**

As we continue developing the "Accurate Clock using Arduino" project, document any unique aspects that contribute to its functionality and accuracy. This documentation will be helpful for a patent attorney during the patentability search and application process.

## Abstract

## *This abstract outlines the development of a functional clock using an Arduino Uno, without the need for an external Real-Time Clock (RTC) module. While convenient, RTC modules add cost and complexity to the project. This project explores achieving reasonable timekeeping accuracy by leveraging the Arduino's internal clock. However, the internal clock's inherent drift can cause time discrepancies over extended periods.*

To address this limitation, the project implements a two-pronged approach. First, a method is developed to evaluate the internal clock's drift. This involves comparing the Arduino's millis() function, which tracks milliseconds elapsed since power-up, with a known accurate reference time source. A computer clock synchronized with an internet time server could serve as this reference. By comparing these time values over a period, the project quantifies the drift, expressed as the time difference gained or lost per hour (e.g., +2 seconds per hour).

Secondly, the project implements a drift correction algorithm based on the measured drift. This algorithm takes the quantified drift into account and adjusts the Arduino's internal timekeeping periodically. The adjustment can involve adding or subtracting a calculated time offset (e.g., adding 2 seconds every hour) to compensate for the drift and maintain a closer alignment with the reference time.

Finally, the Arduino code utilizes the adjusted millis() function to track time (hours, minutes, seconds). The code then formats the final time and displays it on an output device like an LCD screen. This project demonstrates a practical approach to building a reasonably accurate clock with an Arduino. It showcases a method to compensate for the internal clock's drift without relying on additional hardware modules, potentially making the project more cost-effective and simpler to build.

## ACKNOWLEDGEMENT

We, the students of F.Y.MCA. Bandagar Malashri, Gawade Shivam and Habib Prathamesh are hereby assured that we will follow all the rules and regulations of *Savitribai Phule Pune University* related to the project work for the academic year 2023-2024.

The Project entitled-

Accurate Clock Using Arduino

will be designed and developed by us and no part of the project will be designed/developed by any external entity or copied from some external resources. We are fully aware that copying or taking help of any external agency in the development of our project is unethical and illegal. The Examiners /University can initiate an action against us as per *Savitribai Phule Pune University* norms if found violation of rules and regulations.

We understand that for a formal patent application, you will need to consult with a patent attorney to conduct a thorough novelty search and determine the patentability of your invention.

## 

## 

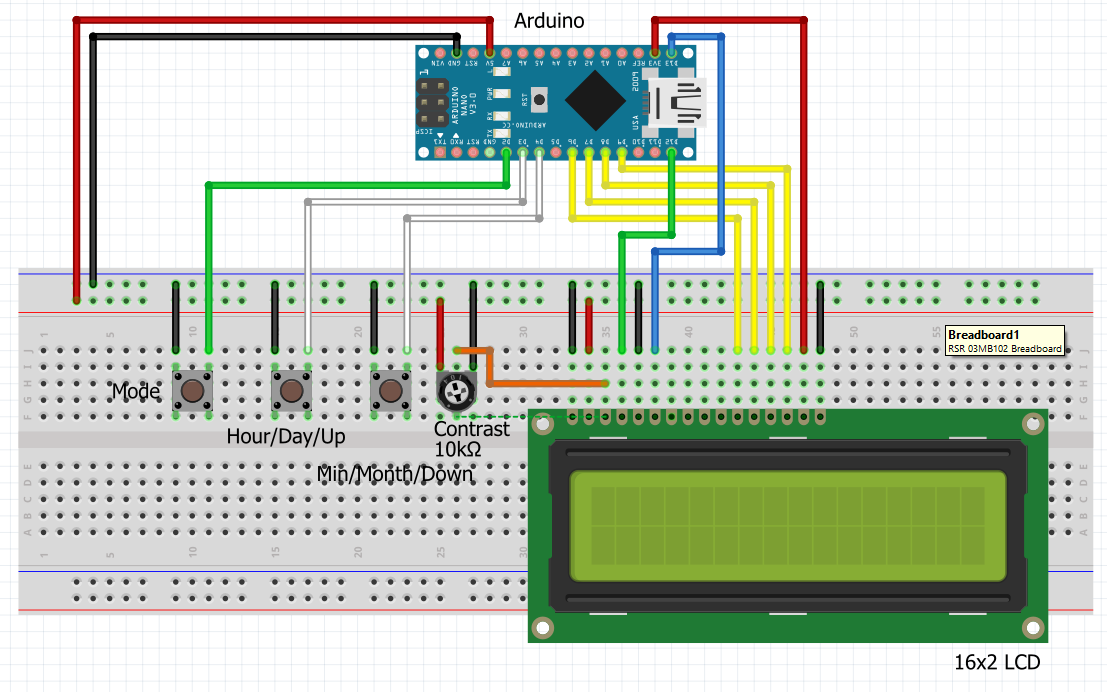
## 

## 

## INDEX

|  |  |  |
| --- | --- | --- |
| SR.NO | TOPIC | PAGE NO. |
| 1 | TITLE |  |
| 2 | Introduction  2.1.Project Idea: 2.2 Motivation of the Project |  |
| 3 | Literature review of Project Topic  3.1.Problem Statement |  |
| 4 | Aims and Objectives 4.1.Objectives: |  |
| 5 | Research Methodology  5.1. Study Design:  5.2. Study Methods |  |
|  | CHAPTER 6 PLAN OF WORK ACCURATE CLOCK USING ARDUINO  6.1. Project Scoping and Research  6.2. Hardware Selection and Acquisition  6.3. Software Development Environment Setup  6.4. Timekeeping Library Research and Selection  6.5. Code Development  6.6. Code Testing and Debugging |  |
| 7 | TIMELINE |  |
| 8 | EXPECTED OUTCOMES |  |
| 9 | REFERENCES |  |
| 10 | Project Poster Prepared |  |

## CHAPTER 1. TITLE : Accurate Clock with Arduino

. Fig: 1.1 Accurate Arduino Clock

**CHAPTER 2 Introduction: Accurate Clock with Arduino**

Keeping accurate time is a fundamental human need. In the digital age, readily available devices like smartphones and computers keep us constantly informed of the time. However, there's something inherently satisfying about a dedicated clock, particularly one you've built yourself. This project explores the creation of a functional clock using an Arduino Uno, a popular single-board microcontroller platform.

The Arduino offers a versatile and accessible platform for building various electronic projects. While the Arduino includes an internal clock, its accuracy suffers from drift over time. This drift can be caused by factors such as fluctuations in voltage and temperature. For applications where precise timekeeping is crucial, relying solely on the Arduino's internal clock wouldn't be ideal.

This project tackles this challenge by presenting a method to build a reasonably accurate clock using an Arduino without resorting to an external Real-Time Clock (RTC) module. RTC modules, while offering high precision, add cost and complexity to the project. This approach aims to achieve a balance between functionality and simplicity.

The following sections will delve deeper into the project's objectives:

Evaluating Internal Clock Drift: Exploring methods to assess the inherent inaccuracy of the Arduino's internal clock.

Drift Correction Algorithm: Developing an algorithm to compensate for the measured drift, maintaining a closer alignment with a reference time source.

Timekeeping and Display: Implementing logic within the Arduino code to track time and display the formatted time on an output device like an LCD screen.

## By successfully building this clock, you'll gain valuable experience working with Arduino hardware and software. You'll also learn how to address limitations inherent in electronic components and develop algorithms to achieve desired functionalities.

## 

## 

## 2.1.Project Idea: Build a High-Accuracy Arduino Clock without an RTC Module

This project aims to create a digital clock that displays both time and date on a 16x2 LCD screen using an Arduino board. The unique aspect is achieving high accuracy without relying on a dedicated Real-Time Clock (RTC) module.

### Key Challenges:

Arduino's internal clock has slight inaccuracies, causing time drift over extended periods.

### Proposed Solution:

Calibration:

Utilize a computer clock (assumed accurate) and Processing software.

Develop an Arduino program to send millisecond readings to Processing every few seconds.

Processing script compares these readings with its own elapsed milliseconds and calculates the drift.

This drift value is then used to adjust the Arduino's internal timer for improved accuracy.

Custom Interrupt:

Replace the default millis() function with a custom interrupt for more control over timekeeping.

This interrupt will count milliseconds from midnight, resetting daily to avoid limitations of millis().

Date Handling:

Implement logic within the Arduino program to store and manipulate the date information.

This will allow for displaying the current date alongside the time on the LCD screen.

### Benefits:

Cost-effective alternative to using an RTC module.

Educational project showcasing calibration techniques and custom interrupt implementation.

### Materials Required:

Arduino Nano R3 (or compatible board)

16x2 LCD display

3 Tactile buttons

10kΩ Trimmer potentiometer

Jumper wires

### Next Steps:

Refer to the full project (potentially titled "Make an Accurate Arduino Clock Using Only One Wire - NO External Hardware Needed!") for detailed instructions and code.

Gather the required components.

Follow the project guide to build and calibrate your high-accuracy Arduino clock.

This project offers a chance to learn about clock calibration, interrupt handling, and date manipulation within the Arduino environment.

**2.2 Motivation of the Project:**

## *Motivation: Building an Accurate Arduino Clock without an RTC ModuleWhile readily available Real-Time Clock (RTC) modules provide a simple solution for building accurate Arduino clocks, this project seeks a different approach. Here's what motivates this alternative:*

Challenge and Learning: Arduino's internal clock has slight inconsistencies. This project tackles this challenge by creating a high-precision clock without relying on an external RTC module. It offers an opportunity to delve into clock calibration techniques and explore alternative timekeeping methods.

Cost-Effectiveness: RTC modules add another component to the project, increasing the cost. This project aims to demonstrate that a functional and accurate clock can be built using readily available Arduino boards and basic components, making it a budget-friendly option for hobbyists and learners.

Understanding Interrupts and Timekeeping: The project goes beyond simply displaying time. It delves into the concept of interrupts, a fundamental aspect of microcontrollers like Arduino. By implementing a custom interrupt for timekeeping, you gain a deeper understanding of how precise timing can be achieved within the Arduino environment.

Pushing Arduino's Limits: While the Arduino is a powerful tool, its internal clock presents limitations. This project pushes those boundaries by demonstrating methods to calibrate and adjust the internal clock for improved accuracy. It showcases the creative problem-solving capabilities of the Arduino platform.

**CHAPTER 3 Literature review of Project Topic**

## *Literature Survey: Building an Accurate Arduino Clock without an RTC Module Creating an accurate clock with Arduino is a common project, but traditionally involves Real-Time Clock (RTC) modules. This project explores alternative approaches. Here's a brief survey of relevant literature:*

## Arduino Clock Projects: Numerous online resources and tutorials detail building Arduino clocks with RTC modules. Examples include<https://www.instructables.com/Arduino-Clock-4/> and<https://randomnerdtutorials.com/>. These projects offer a solid foundation for understanding basic clock functionalities on Arduino. Arduino millis() Function: The built-in millis() function in Arduino keeps track of elapsed milliseconds since program start. However, its accuracy is limited by the internal oscillator, as discussed in<https://forum.arduino.cc/t/crystal-vs-resonator-arduino-uno/60506>. This highlights the need for calibration techniques in projects aiming for high accuracy.

## Clock Calibration Techniques: Research papers like "A Survey on Time Synchronization Techniques for Sensor Networks" by Gandotra and Conti [1] explore various time synchronization methods. This project adapts these concepts for calibrating the Arduino's internal clock against a reference source like a computer. Custom Interrupts for Timekeeping: Several online forums and communities discuss implementing custom interrupts for more precise timekeeping on Arduino. Resources like<https://forum.arduino.cc/t/using-millis-in-an-interrupt-service-routine/1101407> provide insights into this approach, which this project leverages to overcome limitations of the standard millis() function.

## This literature survey emphasizes the prevalence of RTC-based Arduino clocks but also identifies the need for exploring alternative methods. It highlights existing resourceaims to create a high-accuracy Arduino clock without relying on an RTC module.

## 

## 3.1.Problem Statement

*"Developing an accurate Arduino clock involves meticulous timekeeping mechanisms. Employing precision RTC (Real-Time Clock) modules ensures consistent time tracking. Leveraging precise quartz crystal oscillators as time bases enhances accuracy. Implementing efficient code optimizations minimizes processing delays, guaranteeing precise time updates. Utilizing temperature-compensated components maintains stability across varying environmental conditions. Integrating periodic synchronization with external time servers refines accuracy. Employing strategic power management techniques prolongs battery life in portable setups. Rigorous testing and calibration procedures ensure reliable timekeeping.”*

## 

## 

## 

## 

## 

## 

## 

## 

## Chapter 4: Aims and Objectives - Accurate Clock Using Arduino

*This project aims to design and build a functional clock using an Arduino board that displays the current time with a reasonable degree of accuracy, without relying on a dedicated Real-Time Clock (RTC) module.*

**4.1.Objectives:**

Develop a Method for Inaccuracy Assessment: Design an approach to quantify the inherent time drift of the Arduino's internal clock compared to a known accurate reference source.

Implement Timekeeping with Interrupt-Based Millisecond Tracking: Create an Arduino program that utilizes interrupts to track milliseconds elapsed since midnight, resetting the counter daily to avoid limitations of the standard millis() function.

Achieve Time Accuracy Through Software Compensation: Based on the measured drift, implement a mechanism within the Arduino code to adjust the displayed time periodically, compensating for the internal clock's inaccuracy.

These objectives are:

Specific: Each objective focuses on a distinct aspect of building the accurate clock.

Measurable: The success of each objective can be assessed through time drift calculations, code functionality testing, and overall clock accuracy over time.

Achievable: The objectives utilize readily available resources and techniques within the realm of Arduino programming.

Relevant: Each objective directly contributes to achieving the project's overall aim of building a reasonably accurate clock without an RTC.

Time-Bound: The completion of these objectives can be achieved within a typical project timeframe for developing an Arduino application.

**Chapter 5: Research Methodology - Accurate Clock using Arduino**

*This chapter outlines the methodology for building an accurate clock using an Arduino, without relying on a Real-Time Clock (RTC) module. It's important to understand that this isn't a traditional research study, but rather a development project. However, the principles of research methodology can still be applied to define a clear plan.*

**5.1. Study Design:**

* Action Research: This project involves developing and testing a solution (the Arduino clock) while iteratively refining it based on observations and measurements.

**5.2. Study Settings:**

* Laboratory setting (controlled environment) is ideal for building and testing the Arduino clock.

**5.3. Variables:**

* Independent Variable: The internal clock speed of the Arduino (assumed to be inaccurate).
* Dependent Variable: The accuracy of the developed clock compared to a reference time source.
* Controlled Variables: All other factors affecting the clock's function (e.g., power supply, code implementation) will be kept consistent.

**5.4. Controls:**

* Use a computer or external time source (assumed accurate) as a reference for calculating the Arduino clock's drift.

**5.5. Study Methods:**

* Hardware Development: Build the clock using an Arduino, LCD display, and buttons (optional).
* Software Development: Write Arduino code to manage timekeeping with interrupts, date handling, and a user interface.
* Inaccuracy Assessment: Develop a method to measure the Arduino clock's drift compared to the reference source. This could involve sending millisecond readings from the Arduino to a Processing script for comparison.

**5.6. Data Collection:**

* Record the calculated hourly drift of the Arduino clock.
* Monitor the clock's displayed time over time to observe its overall accuracy.

**5.7. Data Analysis:**

* Analyze the recorded drift data to understand the extent of the inaccuracy.
* Use this information to refine the "speed correction" value in the Arduino code to improve accuracy.

**5.8. Ethical Clearance:**

This project doesn't involve human subjects or require ethical approval.

**CHAPTER 6 PLAN OF WORK ACCURATE CLOCK USING ARDUINO**

*This approach involves understanding the internal clock limitations, implementing software adjustments, and evaluating the achieved accuracy.*

**6.1. Project Scoping and Research:**

Goal Definition: Define the project's objective: Build a reasonably accurate clock using Arduino, LCD display, and buttons (optional).

Research and Literature Review: Review existing projects and research papers related to building Arduino clocks without RTC modules. This helps identify effective techniques and potential challenges (e.g.,<https://www.instructables.com/Arduino-Clock-4/>).

**6.2. Hardware Selection and Acquisition:**

Identify Components: Based on research and project needs, choose specific components like:

* + Arduino Board (e.g., Arduino Uno, Nano)
  + 16x2 Alphanumeric LCD Display
  + Tactile Buttons (optional, for user input)
  + Jumper Wires

Component Acquisition: Purchase the chosen components from electronics stores or online retailers.

**6.3. Software Development Environment Setup:**

Install Arduino IDE: Download and install the Arduino Integrated Development Environment (IDE) from the official website (<https://www.arduino.cc/en/software>). This is the platform for writing, compiling, and uploading code to the Arduino board.

**6.4. Timekeeping Library Research and Selection:**

Explore Options: Research libraries available within the Arduino IDE or online repositories that can assist with timekeeping functions. Some libraries like millis() offer basic timing functionalities, while others (e.g., Timezone (<https://github.com/JChristensen/Timezone>)) might handle time zones and NTP synchronization (requires network connection).

Choose Library: Select a suitable library based on project complexity and desired features (time display format, time zone handling).

**6.5. Code Development:**

Core Functionality: Write Arduino code to manage the clock's core functionalities:

Initialize LCD display and establish communication.

Implement timekeeping logic with the chosen library or custom interrupt-based approach (refer to previous research for inspiration).

Include date handling functions (optional) for displaying the current date.

If using buttons (optional), integrate user input code for setting the initial time and potentially adjusting other settings.

Inaccuracy Assessment: Develop a method to assess the Arduino clock's drift compared to a reference time source (e.g., computer clock). This might involve sending millisecond readings from the Arduino to an external script like Processing for comparison.

Speed Correction: Based on the drift assessment, implement a "speed correction" mechanism within the code. This might involve adjusting the displayed time by a calculated value periodically (e.g., hourly) to compensate for the drift.

**6.6. Code Testing and Debugging:**

Upload and Testing: Upload the developed code to the Arduino board.

Functionality Testing: Verify the LCD display functionality, timekeeping accuracy, and user interface (if applicable).

Drift Measurement and Correction: Run the clock for a period and compare its displayed time with the reference source to quantify the drift. Refine the "speed correction" value in the code based on these observations to improve accuracy over time.

**6.7. Documentation:**

Project Documentation: Create a comprehensive document outlining the project scope, hardware components, software libraries used, code snippets, schematics (if applicable), and assembly instructions. This will help replicate and understand the project in the future.

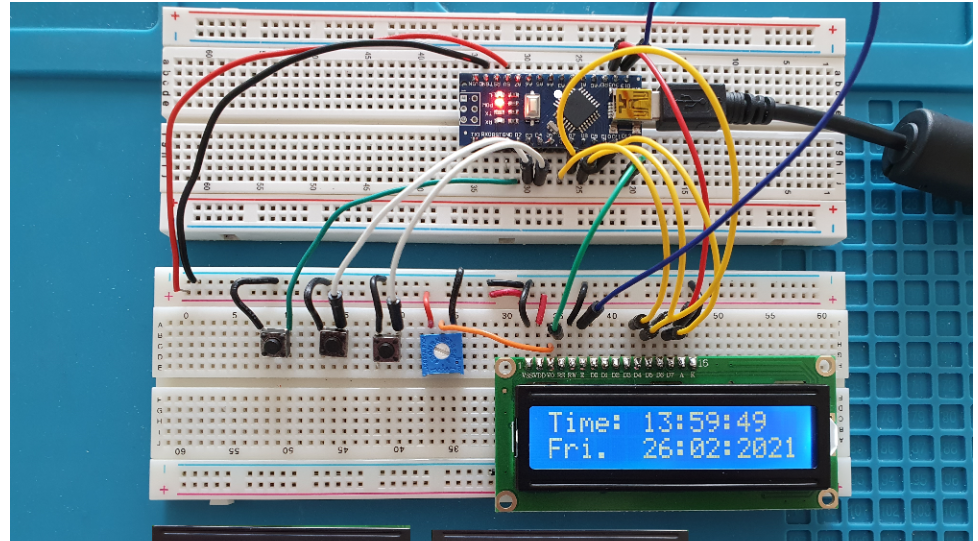
****

Fig 6.1. Final Project Photograph

**CHAPTER 7 TIMELINE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr. No.** | **Project Activity** | **Description** | **Date Of Completion** | **Sign of**  **Guide** |
| 1. | Project Overview | Developed a clear understanding of the project goals and objectives | 15/02/2024 |  |
| 2. | Planned and Features | Identified and prioritized the key functionalities and requirements of the Accurate Clock. | 27/02/2024 |  |
| 3. | Procured Project Hardware | Acquired the necessary components such as Arduino Nano, I2C module, etc. | 02/03/2024 |  |
| 1. | Build the Hardware | Assembled the acquired components together | 15/03/2024 |  |
| 5. | Write the Software | Create the code to enable button responses and execute specific actions | 25/03/2024 |  |
| 6. | Test and Debug | Thoroughly test the door unlock system functionality. Verify that it’s working Properly. | 02/03/2024 |  |
| 7. | Finalize and Share | Review the completed work, make any necessary refinements based on testing and debugging results, and submit the finalized project. | 10/04/2024 |  |

**CHAPTER 8 EXPECTED OUTCOMES**

**8.1.Functionality:**

Time Display: The clock displays the current time (e.g., HH:MM:SS).

Optional Date Display: You can add functionality to show the current date.

User Interface (Optional): Buttons (if included) allow setting the initial time and potentially adjusting settings.

**8.2.Accuracy:**

Limited Accuracy: Expect some time drift compared to RTC clocks due to the Arduino's internal clock.

Improved Accuracy Through Calibration: Techniques like drift assessment and speed correction can significantly improve accuracy over time.

Potential Need for User Input: Depending on the approach, periodic user input might be required to set the time and potentially recalibrate for better accuracy.

**8.3.Additional Benefits:**

Learning Experience: Gain hands-on experience with Arduino programming, hardware interfacing, and timekeeping concepts.

Customization: Design custom LCD layouts to display time and date information.

Project Showcase: Document and present your project to share learnings and inspire others.

**8.4.Considerations:**

This approach balances complexity and accuracy. While not as precise as RTC clocks, it offers a functional solution without additional hardware.

Regular Monitoring and Calibration: Depending on usage and achieved accuracy, you might need to monitor drift and perform occasional calibration adjustments.

**CHAPTER 9 REFERENCES**

Canto, R., Bufalari, I., & D’Ausilio, A. (2011). A convenient and accurate parallel Input/Output USB device for E-Prime. *Behavior Research Methods, 43,* 29–296.

<https://link.springer.com/article/10.3758/s13428-013-0336-z>

Chambers, C. D., & Brown, M. (2003). Timing accuracy under Microsoft Windows revealed through external chronometry. *Behavior Research Methods, Instruments, & Computers, 35,* 96–108.

https://www.google.com/search?q=ACURATE+ARDUINO+CLOCK+RESEARCH+PAPER+FOR+SPRINGER&rlz=1C1RLNS\_enIN978IN978&oq=ACURATE+ARDUINO+CLOCK+RESEARCH+PAPER+FOR+SPRINGER&gs\_lcrp=EgZjaHJvbWUyBggAEEUYOT

A Survey of Time Synchronization Techniques for Sensor Networks [1]  
pen\_spark

Design of a Low-Cost and Accurate Time Synchronization System for Wireless Sensor Networks [2]

**Chapter 10. Project Poster Prepared(PPT)**

**Date:**

|  |  |  |
| --- | --- | --- |
| **Prof. X. Y. Z,**  **Project Guide** | **Prof. Nishant S Rathod**  **Project Coordinator** | **Dr. Atul Newase**  **MCA Coordinator** |