Quest Journals
Journal of Electronics and Communication Engineering Research
Volume 7 ~ Issue 6 (2021) pp: 01-08
ISSN(Online) : 2321-5941
www.questjournals.org

Research Paper

Microcontroller Prototyping Platform with expanded and flexible hardwired I/O resources.

E.C. Anene[1], Abubakar Musa Ibrahim[2],Mahmood Abdulhameed[1]
[1]*Department of Electrical and Electronics Engineering ATBU, Bauchi, Nigeria*
[2]*NCPRD/Energy Commision ATBU, Bauchi, Nigeria*

ABSTRACT
Microcontroller Development platforms are indispensible tools for easy prototyping of microcontroller solutions to electronics projects with control applications. It eases design, simulations, testing and prototyping of standalone or embedded circuits in good times. Limitations of most if not all of these platforms include complexity of application, ease of use and cost. In this paper a platform that is intended to cut down the limitations by offering the inclusion of most common I/O devices on a single board is proposed. Costs and ease of use are to be minimized by featuring libraries that address most common requirements of each I/O resource and make it easy for the user to be able to change or adjust through software the flexible part of the application as needed. Simulations so far carried out showed functionality of the I/O resources. The necessary features to make the platform more flexible are to be added in writing the IDE code.

Received 10 June, 2021; Revised: 22 June, 2021; Accepted 24 June, 2021 *© The author(s) 2021. Published with open access at www.questjournals.org*

I. INTRODUCTION.

A prototype is a first full size functional model of a product to be manufactured. It details the model to be replicated or learned from and serves to provide specifications for a real, working system rather than a theoretical one (*Blackwell, 2015).*The focus of theresearch into creating a prototyping platform is derived from the need to make a circuit board which is easy to use and extensiblewith little effort, cheap, and quick to realize. Use must certainly to be made of widely available development platforms to achieve more simplified platforms in the sense of ease of use and ease of design applications and alsotomake learning of design and production of wide ranging electronics products easier, friendlier and cheaper than Arduino based platform for Industries, Colleges and Universities. In this proposed platform, it is intended that ease of application be obtained through simplification of the IDE resources to enable simple plugin and deployment. This will be done by specific allocation of ports to specific resources which is implemented by routines specifically adapted to it. For example a temperature sensorwill be at a specific location on the port with a corresponding routine to run its application.

Therefore, the new platform mustoffer some balance between cost, expandability, improvement of quality of teaching and ease of use and proffer local solutions in developing economies. This is to lower time implementation needed in application and use in Industries, Colleges and Universities to deploy hardware solutions. To encourage an increase in numbers of developers and thus products, such platforms must meet such basic requirements. Existing platforms like Arduino, Programmer 2003, MicroDig platforms etc do not meet all the requirements. Students and beginners also find it difficult to apply some of their concepts when using those boards because of the specialized nature of implementing the add-on's and basic peripherals.

Such a new Platformshould allow users to create working electronic prototypes, either stand-alone objects or devices tethered to a computer. It should read from a wide range of sensors, control a broad spectrum of output devices, and communicate with software running on a computer over a network etc. The most important features of this proposed Platform are to be: It's ability to accept program from PC; availability of library for all add ondevices that someone may think of connecting e.g sensors, output devices etc; plug/activate features that permit users to plug any I/O devices and provision of expansion slots for future use is proposed.

II. LITERATURE REVIEW.

This Literature presents a survey of a few of the more popular microcontrollerboard development tools on the market, followed by an analysis of how their strengths and weaknesses affect design choices. At the highest level of abstraction are microcontroller tools such as Infusion Systems' MicroDig, Phidgets, and Stanford's d.tools (Michael, 2013). Modules at this level are generally not programmable by the end user.

Instead, they are configured using adesktop tool. These tools are generally not standalone devices, but must be connected to a personal computer in order to be useful. Infusion Systems' MicroDig(Reas, 2005) is a sensor interface box with a MIDI interface. It's hardware interface consists of an analog-to-MIDI controller with 8 analog inputs, and various sensor modules that mate with the controller. Users attach pre-packaged sensors to the inputs, and connect the controller to a MIDI output device. The values of the sensors are output as MIDI values. The MicroDig is handy for teaching students with some little familiarity of MIDI and programming or knowledge of electronics and how to design hardware interfaces. Nonetheless, it is an expensive platform, with the basic kit costing $399, and requires that the connecting equipment be MIDI compatible.

Phidgets is a modular system of sensor controllers, motor controllers, RFID readers, and other special function devices, all united by a common USB interface and a set of desktop software APIs (Szekely, 1993). Each Phidget device is a self-contained electronic device, whether it's a sensor, motor or LED controller, or a more complex device like an LCD display. The user needs almost no knowledge of electronics to use Phidgets. Each device is connected to a desktop computer in order to access its sensor data or to control it (Adrian et al 2015). The modules arerelatively inexpensive, ranging from $10 to $100 but the devices cannot be used as standalone units, however and must be interfaced to a personal computer to use. The learning curve for Phidgets is somewhat steeper than for the MicroDig, but it's useful for those familiar with software development who want to begin making hardware interfaces.

d.tools is a high-level hardware and software tool developed at Stanford University's HCI group that addresses some of the shortcomings of others in this class(Wixom et al 2005). First, d.tools is a more flexible system. The d.tools software can be used with other hardware platforms, as long as that hardware is running a firmware that can communicate in the d.tools protocol. Wiring, Arduino and Phidgets hardware have been used with d.tools. The software is written in Java as a plugin for the Eclipse universal tool platform, and can theoretically run on any Java-capable operating systemtools in an open source platform(Adrian et al 2015).There are many other microcontroller development tools available for use in teaching and prototyping. Those that are popular outside the electrical engineering community offer some balance between cost, expandability, and ease of use like Arduino withhardware made up of a series of plug-and-play USB modules(Adrian et al 2015).

Moving down a level of abstraction, there are a number of mid-level microcontrollers featuring basic support electronics (crystal, power regulator, etc.) on a small module. These modules are built with the assumption that users can build inputand output circuits to attach to it. They're usually programmed in BASIC, or some variation of C, and attached to the programming environment on a personal computer using a serial or USB connection. Parallax' BASIC stamp is the most well-known of these modules (Michael, 2013). Also in this family are NetMedia'sBasicX , BASIC Micro's Basic ATOM processors.The main advantage offered by this range of controllers is programmability in a high-level language, with a simple programming interface. The disadvantage is generally that the programming languages are very limited, and the lower levels of the controller itself are not accessible to the user at all. Students taught to use these controllers generally reach a problem beyond the module's capability within their first semester. The programming environmentsare almost all available for Windows operating systems only, though there are some exceptions.The processors themselves are usually priced around $50, and experimenter's kits, including a processor, power supply and prototyping board usually cost between $75 and $100. Since they're aimed at beginners, this price, while less than the high-level controllers, is still high. Users do not often think about using multiple modules in a project because the price of multiples is prohibitive. With mid-level microcontrollers, the mistake of wrong wiring of a circuit and destroying the processor is high.

At the lowest level of abstraction are the microcontrollers themselves. Two of the most popular are Microchip's PIC family of processors, and Atmel's AVR processors. These are programmed in C or Assembly or BASIC, and much of the programming involves direct access to the processor's registers. A separate hardware programmer is usually needed tocommunicate between the programming environment and the processor. The developer must build the necessary support circuitry for the processor in addition to any sensor or actuator circuits. The learning curve for this class of controllers is the steepest of any mentioned here.

The advantage offered by lower level controllers is cost. At $1 - $15 apiece, they can be used in multiples easily. There is often some initial setup cost for the development environment, however. Programmers range from $30 - $100, and more fully-featured programming environments can be in excess of $300. This cost is amortized by continued use: the more processors you use, the cheaper it gets. There are some good open source development environments, particularly for the AVR controllers, but they are not designed for beginners. The interfaces are usually complex, or command-line interfaces, and the code libraries require a working knowledge of C or C++. (*Blackwell, 2015)*

Finally there are Arduino's predecessors, Programma 2003 and Wiring. One of the authors (Reas, 2005) developed in 2003 a simple micro-controller platform called "Programma 2003" based on a PIC chip and the open source language Jal. The design goal for this platform was to have something as cheap as possible which would be open source and run on Windows, MacOS X, and Linux. Programma 2003, however, lacked

good documentation, a wide community, and used a relatively unknown programming language that lacked certain key features.Wiring is a mid-level module, based on one of the AVR microcontrollers whichattempt to address the programming interface limitations of the families of processors above.

The programming environment for Wiring, on which the Arduino environment is based, has its origins in Processing (Reas,2005), a multimedia programming environment. Wiring, like Processing, was made to teach design students about programming. The environment itself is simple to understand. The language uses clear terms for command names like analogRead() and digitalWrite() rather than the more terse style usually associated with microcontroller flavors of C. The interface for uploading programs to the microcontroller is minimal, allowing users to focus on the task of programming. Menu item names are unambiguous, and kept to a minimum. The environment is written in Java, and is available for Windows, OSX, and Linux, unlike most other microcontroller development environments. Its availability for OSX alone has led to its increased use by designers and artists who prefer that platform. The Wiring module is a powerful tool, but it is limited in that it cannot be easily used by a beginner. The module as sold by the developers is priced in the range of other mid-level modules, around $60 -$80, too expensive to allow for easy experimentation or use ofmany boards.

The proposed Platform will be designed to some extent to resemble Arduino in the way of programming but cheaper and more flexible for ease of use.It is intended that added features will make it more user friendly and provision for input sensors like DHT22, GPS Module, etc made. This Platform is expected by design to afford students the ease and flexibility of using add-ons in basic slots provided for them without complicated circuitry contrary to when users need to do bulky connections on bread board before demonstration onArduino Platform. Such will also be useful for advanced learners and researchers to achieve rapid prototyping.

III. MATERIALS AND METHODS.

Development of such platforms first requires conceptualization of the idea based on the identified problems and literature gaps. Materials needed spring out of such conceptualization leading to designs and evaluation of designs. Simulations are done on the designs after which the prototype is constructed. For this Platform, ease of use will be achieved by specific allocation of ports to specific resources. To make all connections possible, it is intended that the ports be hardwired on the boardto make all communications possible from as simple as LED up to complex communications like Wi-Fi, GSM etc.

This platform will feature a microcontroller (control unit), connected to resource ports. The microcontroller intended for use is the ATmega 328. ATmega328 has three types of memory: **Flash memory**-32KB nonvolatile memory for storing application, **SRAM memory**-2KB volatile memory for storing variables used by applications while it's running, **EEPROM memory**-1KB nonvolatile memory used to store data that must be available even after the board is powered down and then powered up again (Tawil 2016).

The I/O resources include: UART interface (link between PC and Microcontroller to upload instructions) and the following dedicated and reconfigurable ports: I^2C, PWM, SPI, Boolean, Analog, USB interface.It will also have indicators, Clock and Power supply. For this work CH341 chip will be used as a USB for UART interface because it cheap and available, supports full speed USB interface and is compatible with USB 2.0 interface. It supports all existing applications using serial ports without the need of changing existing code as well as supports 5volt and 3.3volt operation and common flow control signals. The concept of the proposed system stems from the plan to have a low cost, easy to use platform. This implies that the I/O resources must somehow be made easily accessible and programmable. Figure 3.2 shows the concept plan having the commonly available I/O devices represented.

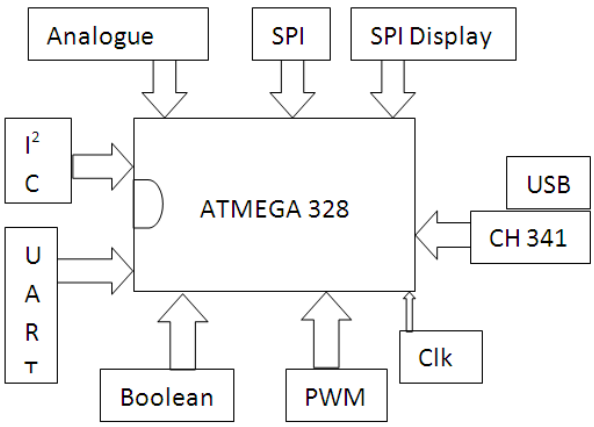


Figure 3.2 Platform Concept plan

3.1. Methodology

The methods used in the work flows from the Concept plan to series of vetting of proposed ideas then to design and connection and simulations. All are done on the Proteus development software using the Ardiuno IDE platform. The Ardiuno IDE is used to minimize platform development time. Specific IDE for the proposed platform will be developed when it is up and running. Figure 3.1 show the flow of the methods applied.

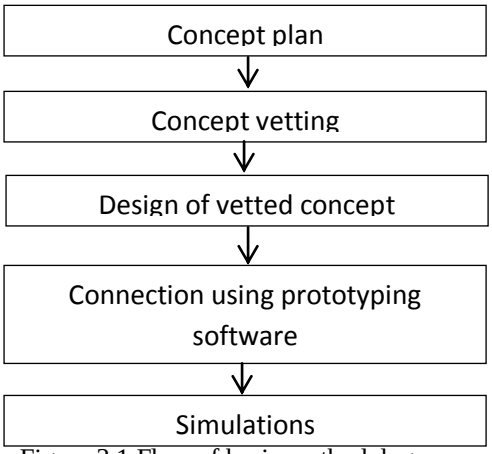


Figure 3.1 Flow of basic methodology

IV. PLATFORM DESIGN.

The circuit was connected on the Proteus platform.The proposed device I/O ports were wired on the ATmega 328 microcontroller. Appropriate connectors were placed for the proposed devices according to the design layout using a combination of drill through and surface mount options. Connection errors were detected and corrected. Plate 1 shows the resulting schematic.

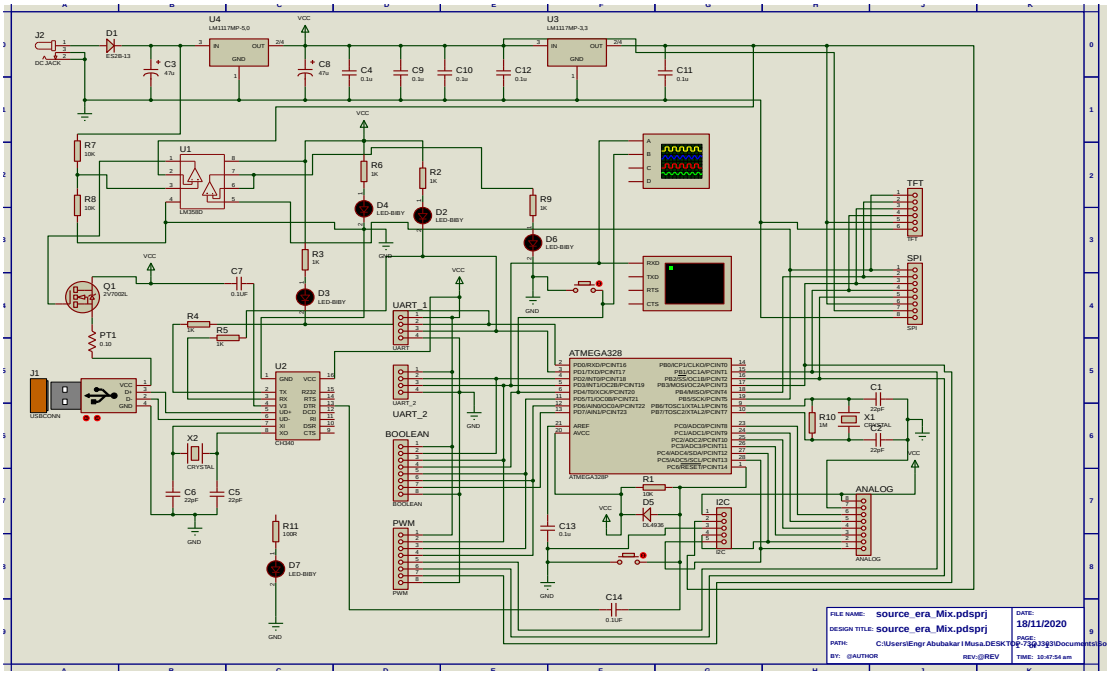


Plate 1: Circuit Schematic

Using Proteus diagnostic resources the schematic was translated into the board layout by placing the appropriate devices from the Proteus libraries. Plate 2 shows the board wiring layout and the layout depicting the proposed board I/O resources respectively. Plate 3 shows the labeled resources including: UART 1, UART2, SPI, TFT, I^2C, Boolean, Analog, PWM.

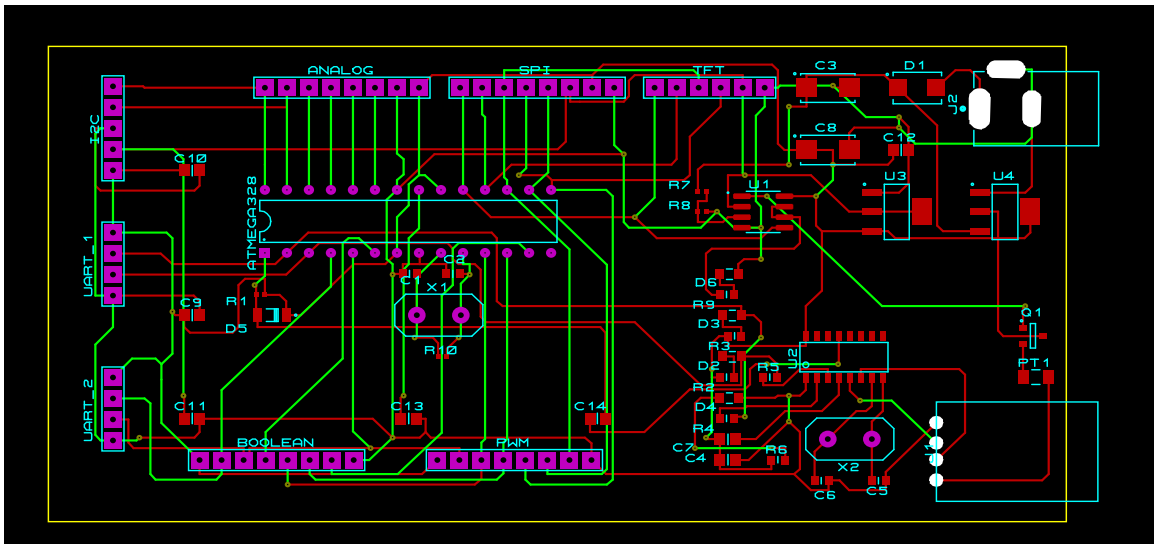


Plate 2. The proposed board wiring layout

Plate 4 shows the 3 D view of the board.

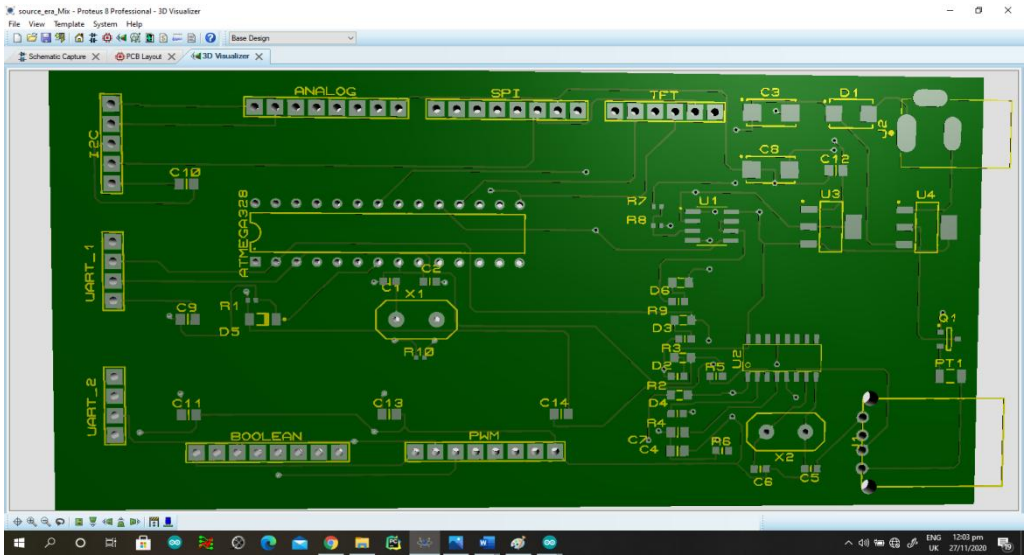


Plate 3: Board Layout showing the board resources.

Plat 4: 3D rendition of the proposed board

4.1 Simulation.

To simulate the new design, Ardiuno IDE was used at the frequency of 16MHz and 9600 Baud rate. The ATmega 328 default frequency is 8MHz and so the system timing must be software programmed for external timing control. Figure 4.1 shows a snippet of the program that simulates UART 1. It is done by making

a typical UART device receive information from the CPU and send to the monitor. UART 2 is also simulated similarly.

UART_1

```
Program Code generated with Arduino IDE
The RXD/TXD of the ATMEGA328 is at the default (D0 = RXD and D1 = TXD)
void setup() {
  // initialize both serial ports:
pinMode(3, INPUT_PULLUP);
Serial.begin(9600);
Serial.println("Sorce Era Serial");}
void loop() {
// read from port 0, send to port 1:
if(!digitalRead(3)){
Serial.println("Button Pressed");
while(!digitalRead(3)){delay(100);}//debounce
  }
if (Serial.available()) {
intinByte = Serial.read();
Serial.write(inByte);
```

Figure 4.1 Snippet of UART 1 simulation

V. RESULTS AND DISCUSSIONS

Simulation results are presented from the input and output responses captured from thevirtual terminal and digital oscilloscope in the Proteus software. Voltages and time data are also obtained from it.

5.1 Simulations Results.

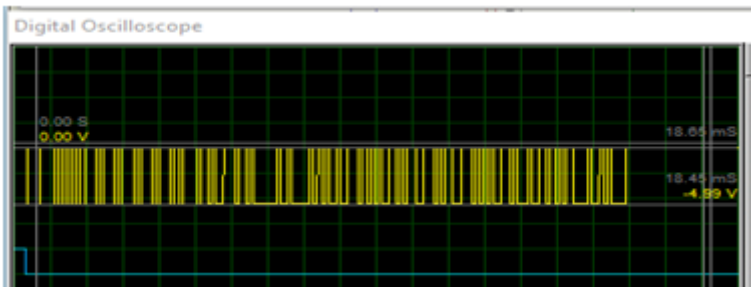
5.10 UART

To simulate the UART, measurements of sending time and voltage were measured from the Digital Oscilloscope.

Sending time 0.00ms-16.30 ms, Voltage 0-4.99V

Plate 5 shows the digital write and the corresponding signal waveforms for the instruction "Button Pressed"

b)



a)

Plate 5a) Instruction from microcontroller b)Oscilloscope wave formrepresenting the statement.

5.11 PWM

Plate 6 shows the waveform resulting from the PWM code where the analog write port is given a fading signal in an increasing measure.

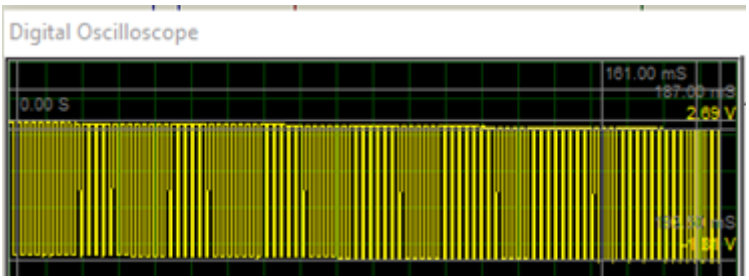


Plate 6 PWM simulated waveform

5.12 SPI Simulation

The SPI stands for Serial Peripheral Interface which is a serial communication and it is used for short distance communications. Its common uses are in sensor, LCDs and Secure digital cards.It uses four (4) cables where 3 of them are the same for each device (MOSI, MISO, SCK) and the fourth one is for selecting the device address. There is a fifth cable is the GND. Plate 7 shows the output waveforms using the SPI debugger. Channel A (yellow) is the SS or chip select, Channel B (Blue) is the MOSI, Channel C (Pink) is the MISO.

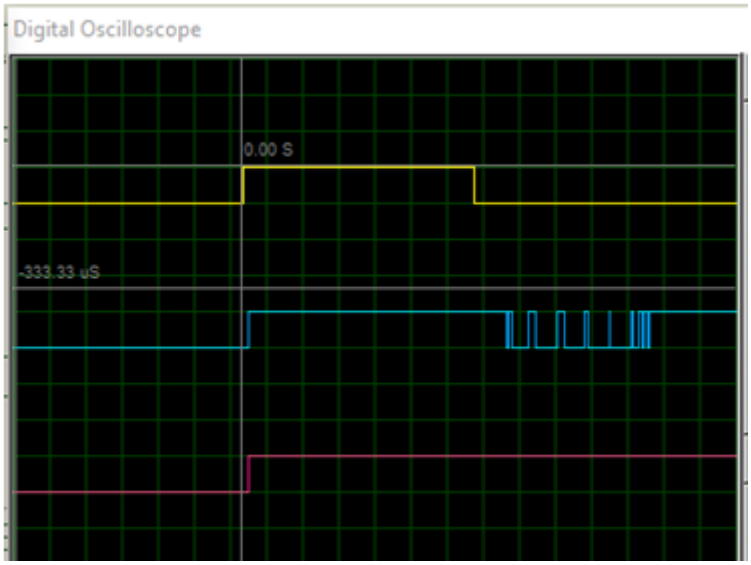


Plate 7 SPI Simulated waveform

5.13 I^2C Simulation

This I^2C protocol is simulated using the debugger. Plate 8 shows theI^2C output waveforms. The lines showSDA which stands for Serial Data Line and SCL which stands for Serial Clock Line.Channel A (Yellow) is SDA andChannel B (Blue) is SCL.

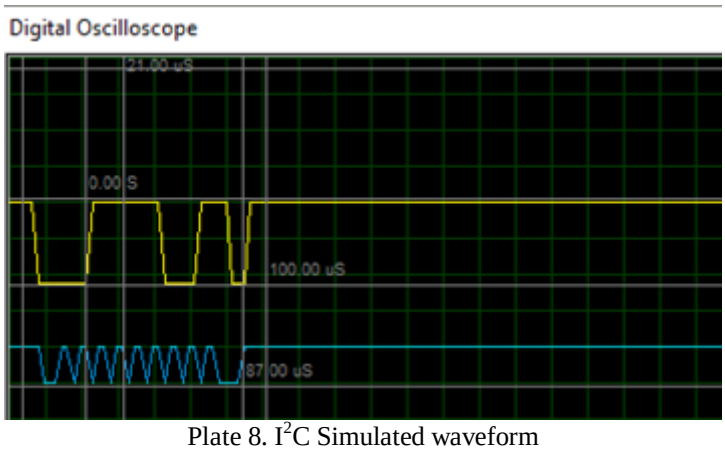


Plate 8. I^2C Simulated waveform

VI. CONCLUSION.

It has been seen that the versatility of the microcontroller design can be employed to simplify its application by making most of its resources dedicated to handle specific interfaces that are most commonly used in practice. This realizes the desire to make it easier to use and expand. More work is being done to provide low cost solutions to the proposed development board.

ACKNOWLEDGEMENT

Special apreciation goes to TETFUND and Abubakar Tafawa Balewa Univeraity Bauchi because This work was carried out through the sponsorship of Tertiary Education Fund (TETFUND) under Institution Based Research (IBR) grant in 2019 through the Abubakar Tafawa Balewa University Bauchi Research and Development Department.

REFERENCES

[1]. Szekely, P., Luo, P. and Neches, R. (1993). Beyond Interface Builders: Model basedInterface Tools. Proceedings of ACM/IFIP Conference on Human Factors

[2]. Blackwell, S Wegner, P. (1997). Why Interaction is More Powerful Than Algorithms.Communications of the ACM, 40(5):80-91.in Computing Systems, INTERCHI'93, pp.383-390

[3]. Reas, Casey and Fry, Ben,(2005) Processing.org: a networked context for learning computer programming." ACM SIGGRAPH, 35-55

[4]. Wixom, B. H., & Todd, P. A. (2005).A theoretical integration of user satisfaction and technology acceptance.Information systems research, 16(1), 85-102.

[5]. Michel, B. Wendy, E. (2013) Prototyping Tools and Techniques. Arduino prototyping Platform , page(1-10).

[6]. Adrian, Y. Thor, W. Sal, T (2015) Introduction to Arduino Prototyping, arduino research and development page(1-17)

[7]. YahyaTawil, "Understanding Arduino UNO Hardware Design"https://www.allaboutcircuits.com/technical-articles//July 2016