E381A - 2023-24
EC LAB PROJECT

Section: C
Table Number: 17

Team Members:
Nishant Patel (210673)
Monkesh Singh (210631)
Mohd. Sibtain (210616)

Q1: What problem are you trying to solve, and why is it important/interesting?

We're developing a component tester to assess the functionality of ICs, detect defects, and measure resistance and capacitance. This tool addresses the common challenge of uncertainty in lab work due to faulty ICs. By providing accurate testing and measurement capabilities, it ensures reliable performance and efficiency in electronics experimentation. Ultimately, it streamlines troubleshooting processes and enhances productivity in lab environments.

Q2: What are the existing solutions? Describe a few of them and list any shortcomings in them. Is your solution approach unique in some way?

Existing solutions like IC tester machines are indeed effective but often come with a high price tag, ranging from 30,000 to 35,000 INR. These machines provide comprehensive testing capabilities but may be cost-prohibitive for many labs or individuals. Our solution stands out by offering similar functionality for a fraction of the cost, making it accessible to a wider audience without compromising on quality or accuracy. Its affordability and effectiveness make it a unique approach to addressing the same problem.

Q3: What resources do you require to complete the project? Give a breakdown of the tasks that you need to accomplish week by week to complete the project.

<u>Week 1: Arduino and Bluetooth Setup and Component Familiarization</u>

- Procure necessary components including Arduino NANO, HC-05 Bluetooth module, breadboard, jumper wires, resistors, and capacitors.
- Prioritize setting up and testing the Bluetooth module to ensure its functionality.
- Familiarize ourselves with the specifications and operation of all components through datasheets and online resources.
- Verify proper functioning of the motion sensor and understand its interfacing requirements with the Arduino NANO board.

<u>Week 2: Circuit Design and Prototyping</u>

- Design a schematic diagram for integrating all components.
- Assemble the circuit on a breadboard, ensuring proper connections and compatibility between components.
- Test individual components to verify functionality and troubleshoot any issues.

<u>Week 3: Software Development and Integration</u>

- Write Arduino code to implement number lock functionality using the keypad.
- Integrate hardware and software components to create the complete Components tester.
- Conduct thorough testing to validate the outputs.

PROJECT REPORT

Project Objective:

The primary objective of our project is to develop a comprehensive component tester aimed at evaluating the functionality of integrated circuits (ICs), identifying defects, and precisely measuring resistance and capacitance values. This tool is designed to tackle the prevalent issue of uncertainty in laboratory settings caused by faulty ICs. By offering precise testing and measurement functionalities, it gives dependable performance and effectiveness in electronics experimentation. Ultimately, our goal is to streamline troubleshooting procedures and elevate productivity within laboratory environments, thereby fostering innovation and advancement in electronic research and development.

Resources Used:

https://docs.arduino.cc/learn/communication/wire/
https://www.arduino.cc/reference/en/language/functions/communication/wire/
https://www.exploreembedded.com/wiki/Setting_up_Bluetooth_HC-05_with_Arduino
https://www.circuitbasics.com/arduino-ohm-meter/
https://docs.arduino.cc/tutorials/generic/capacitance-meter/
https://www.makerguides.com/arduino-and-hc-05-bluetooth-module-complete-tutorial/

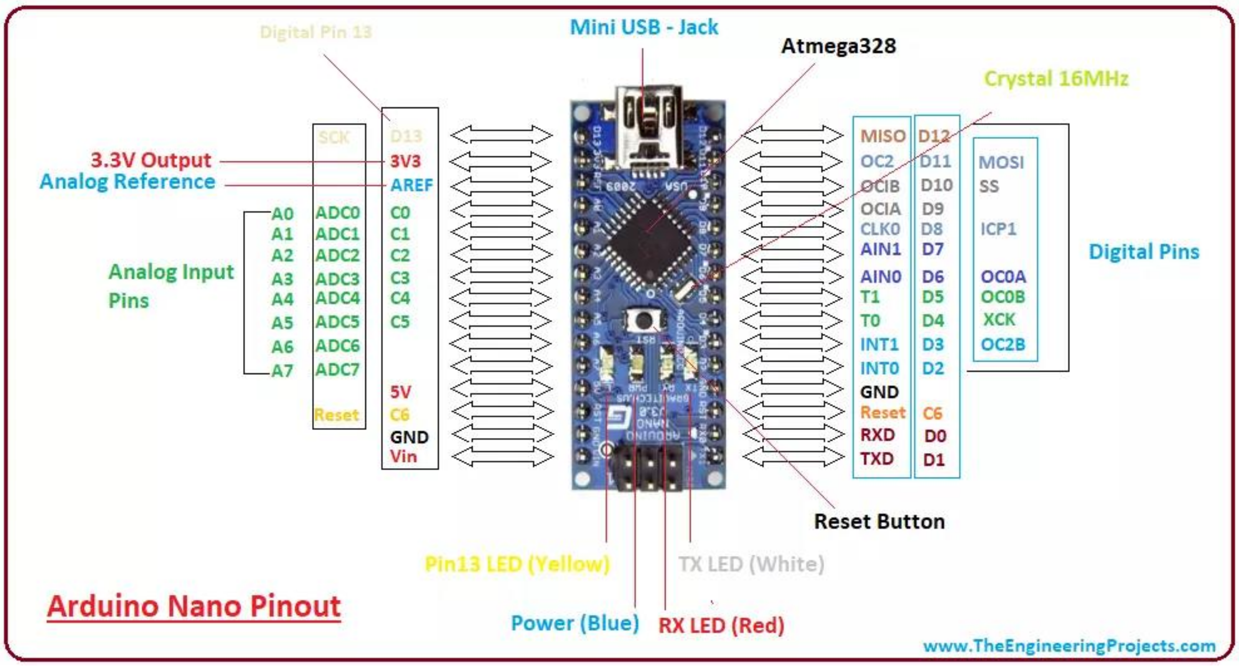
Description of each component:

1. Arduino UNO:

The Arduino Nano is an open-source breadboard-friendly microcontroller board based on the Microchip ATmega328P microcontroller (MCU). It offers the same connectivity and specs of the Arduino Uno board in a smaller form factor.
The Arduino Nano is equipped with 30 male I/O headers, in a DIP-30-like configuration, which can be programmed using the Arduino Software integrated development environment (IDE), which is common to all Arduino boards and running both online and offline. The board can be powered through a type-B mini-USB cable or from a 9 V battery.

Some of the key specifications of the Arduino UNO board are:

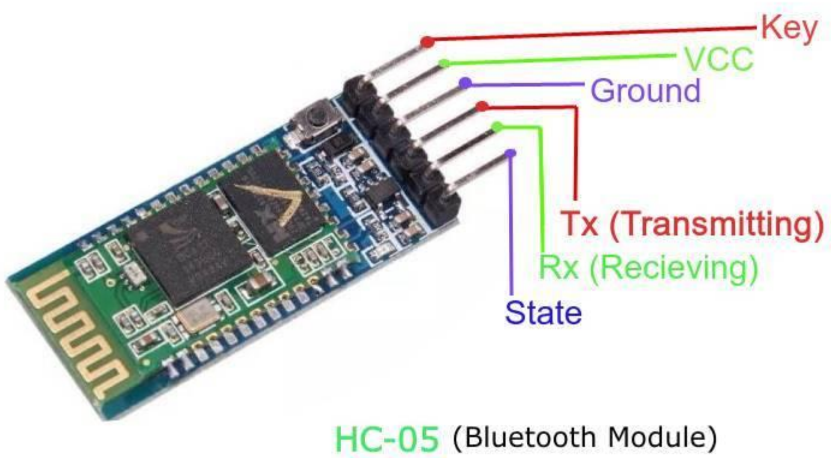
- Microcontroller: Microchip ATmega328P
- Operating voltage: 5V
- Input voltage: 5 to 20 volts.
- Digital I/O pins: 14 (6 optional PWM outputs)
- Analog input pins: 8
- DC per I/O pin: 40 mA
- DC for 3.3 V pin: 50 mA
- Flash memory: 32 KB, of which 2 KB is used by bootloader.
- SRAM: 2 KB
- EEPROM: 1 KB
- Clock speed: 16 MHz
- Length: 45 mm
- Width: 18 mm
- Mass: 7 g
- USB: Mini-USB Type-B



2. HC-05 Bluetooth Module:

The HC-05 Bluetooth module is a popular module that allows wireless communication between devices over Bluetooth. It uses the Bluetooth 2.0+EDR (Enhanced Data Rate) protocol and can be configured as either a master or a slave device. Here are some specifications of the HC-05 module:

- Bluetooth Version: Bluetooth 2.0+EDR
- Frequency Band: 2.4 GHz ISM band
- Modulation: GFSK (Gaussian Frequency Shift Keying)
- Transmit Power: Class 2, up to 4dBm
- Sensitivity: -84dBm at 0.1% BER
- Range: Up to 10 meters (Class 2)
- Operating Voltage: 3.3V DC to 6V DC
- Current Consumption: <30mA (at 3.3V DC)
- Interface: UART (Universal Asynchronous Receiver/Transmitter)
- Baud Rate: Default 9600 baud, configurable up to 1382400 baud
- Dimensions: 28mm x 15mm x 2.35mm

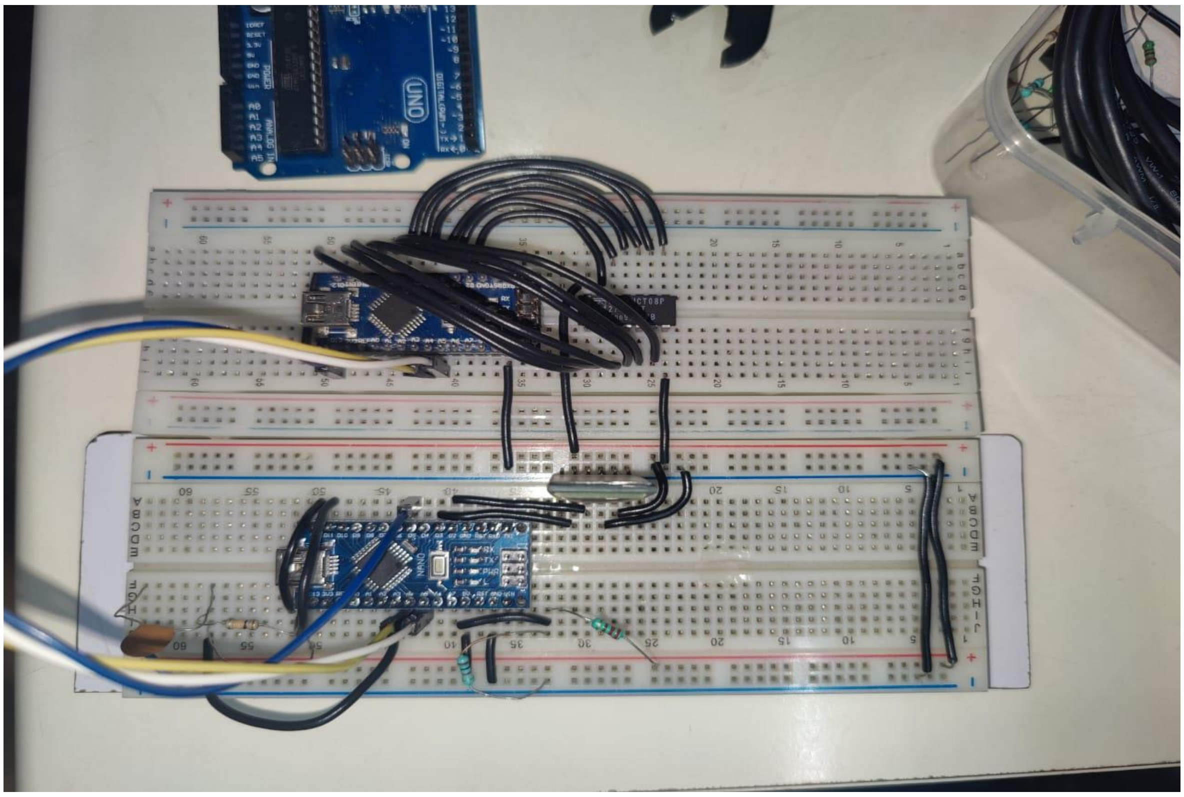


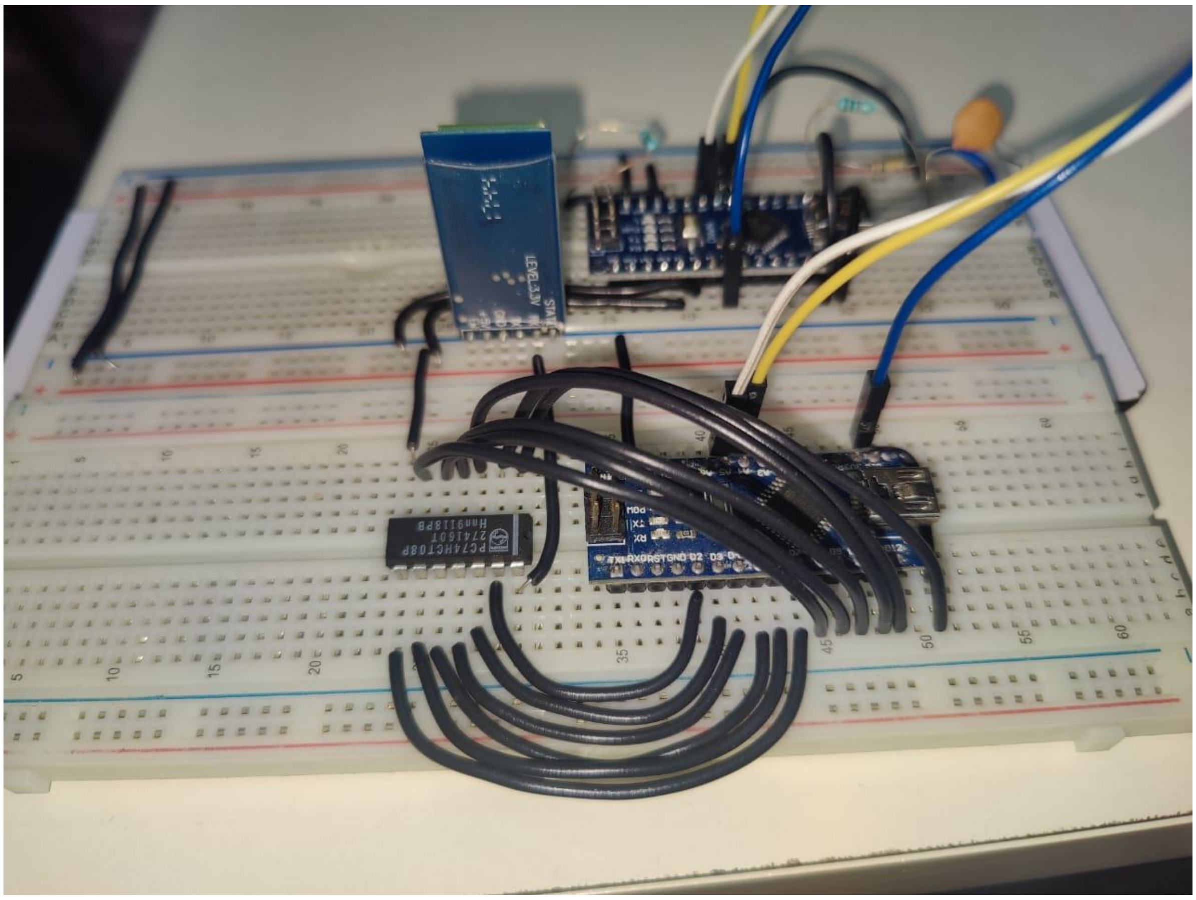
HC-05 (Bluetooth Module)

Cost Analysis:

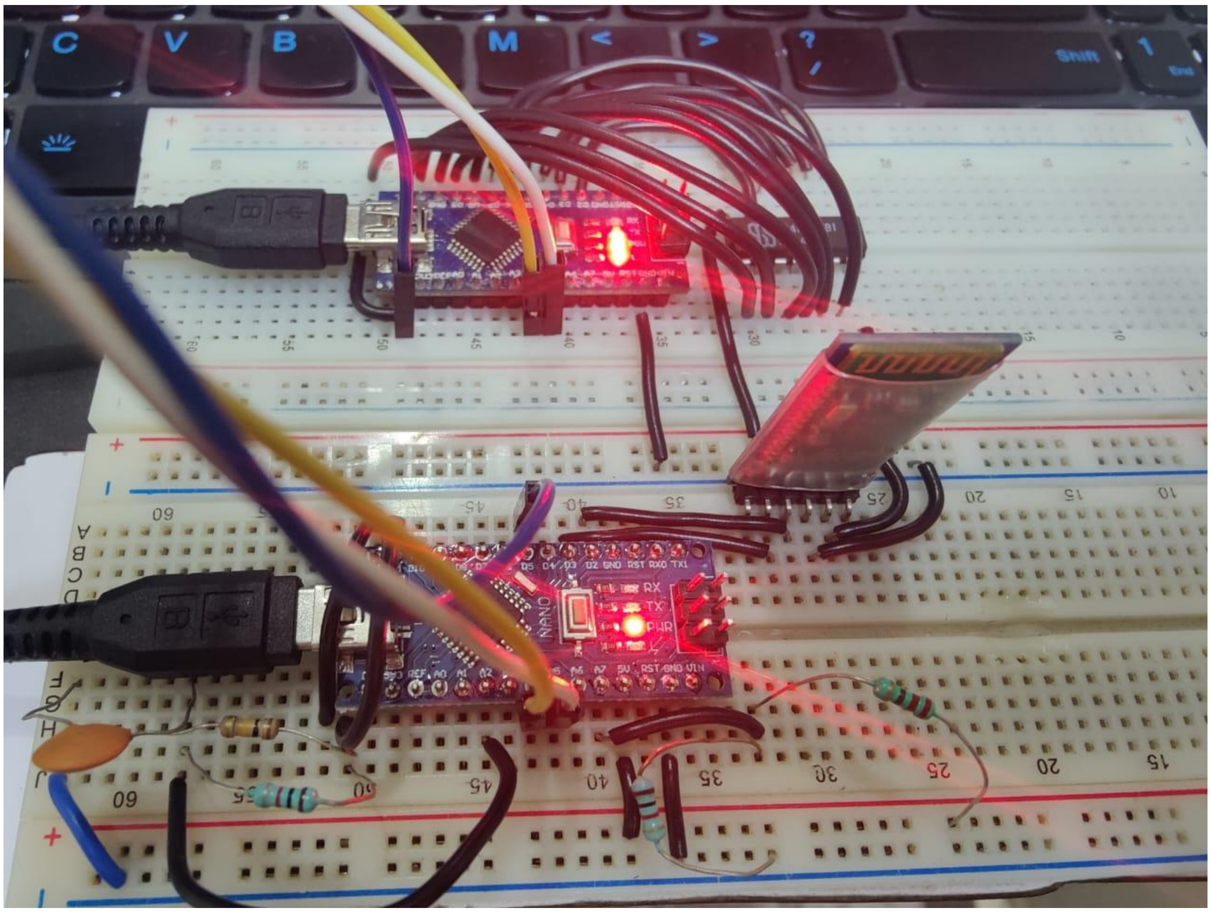
Component	**Cost**
Arduino NANO*2 + USB cable	300*2=600
HC-05 Bluetooth Module	500
Breadboard*2	60*2=120
Jumper Wires	50
Resistances	10
Capacitors	50
Total	**1330**

Circuit Diagram:



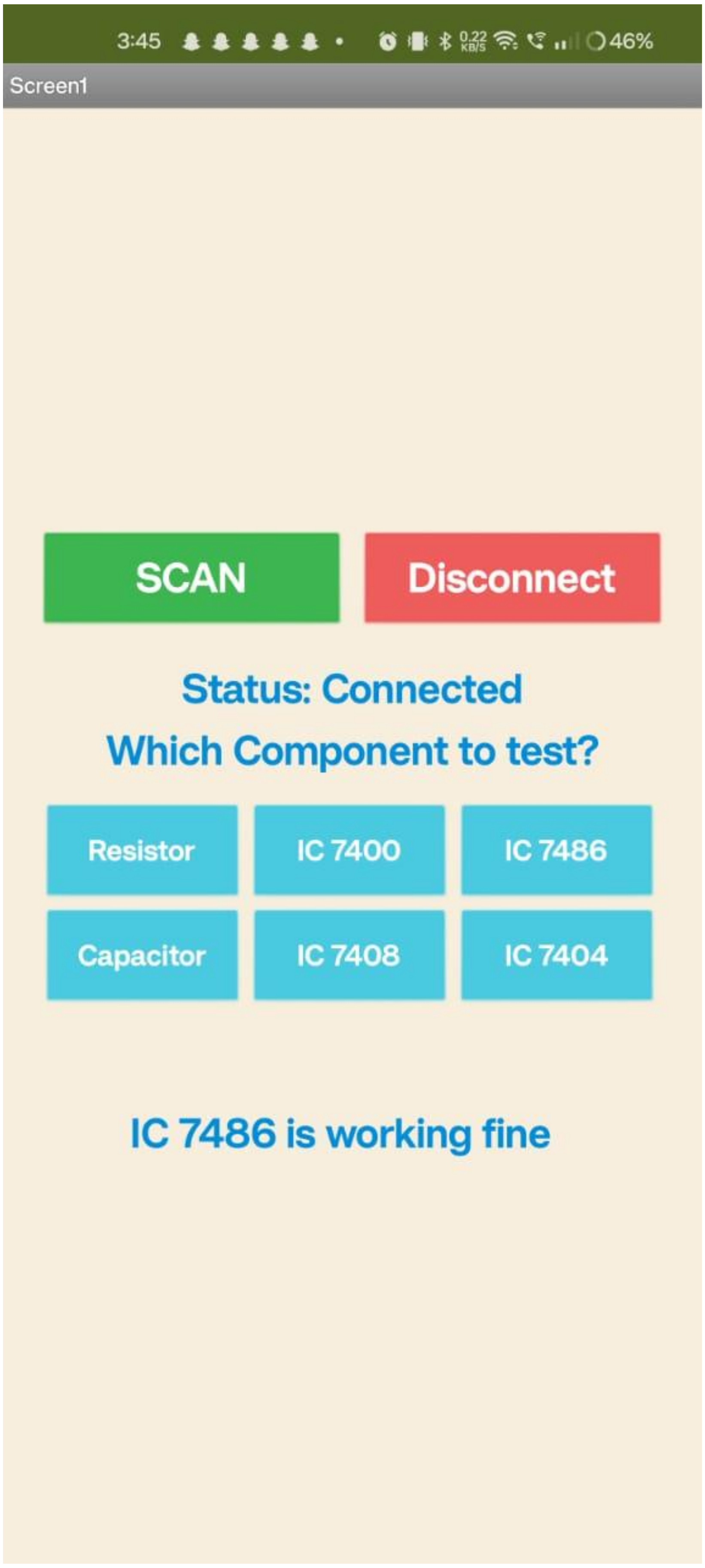


Results:



Final circuit was completed as shown. We also made an application using MIT app inventor (screenshots and codes of that are given below). There is a button provided by the app named SCAN for connecting with HC-05 Bluetooth module. Just below the scan button there are 6 other options provided namely RESISTOR, CAPACITOR, IC 7400, IC 7486, IC7408 and IC 7404. The user can choose the component to be tested. After that, If the user selects any IC then it'll output if it is working fine or not. If the user opts for resistance or capacitor, then it'll output the value of that component.

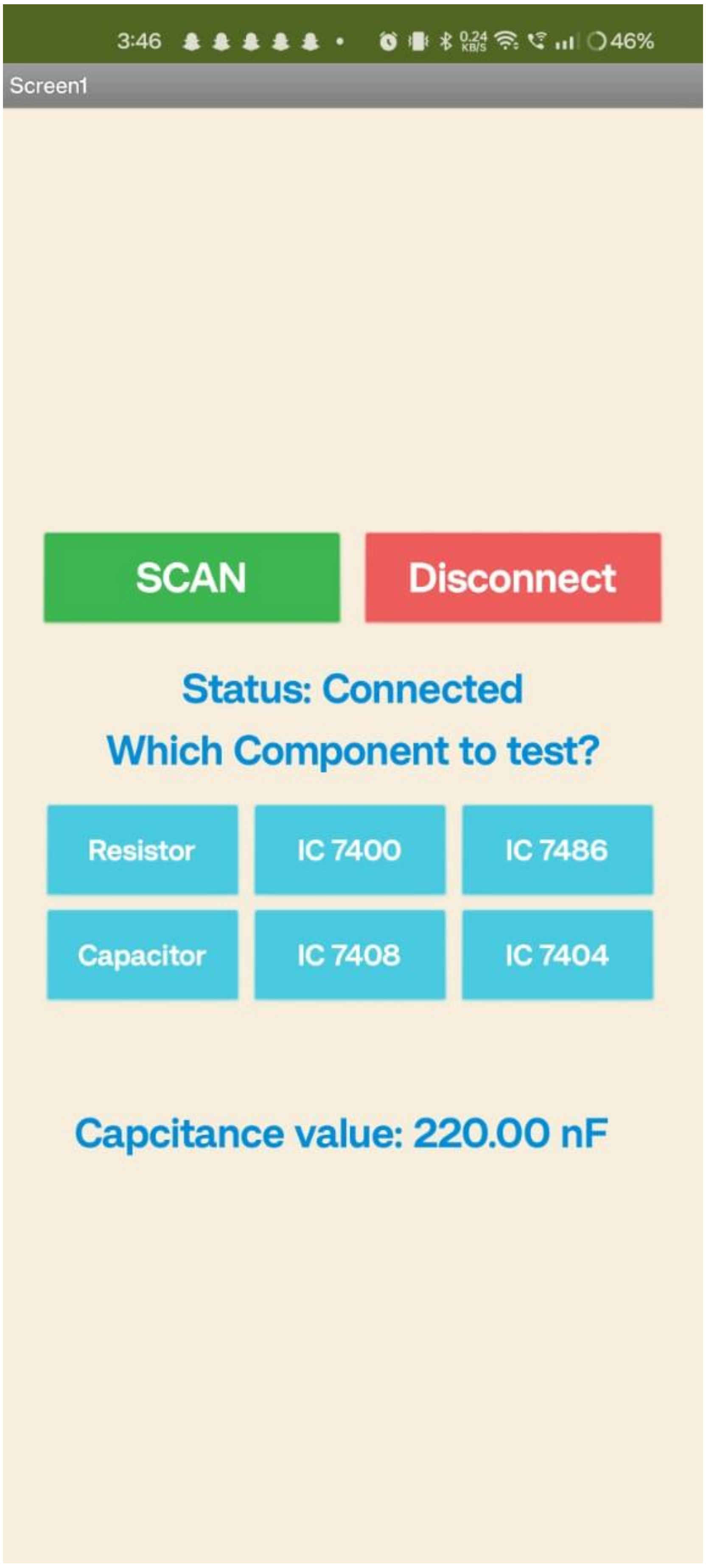
Application UI-



Status: Connected

Which Component to test?

IC 7486 is working fine

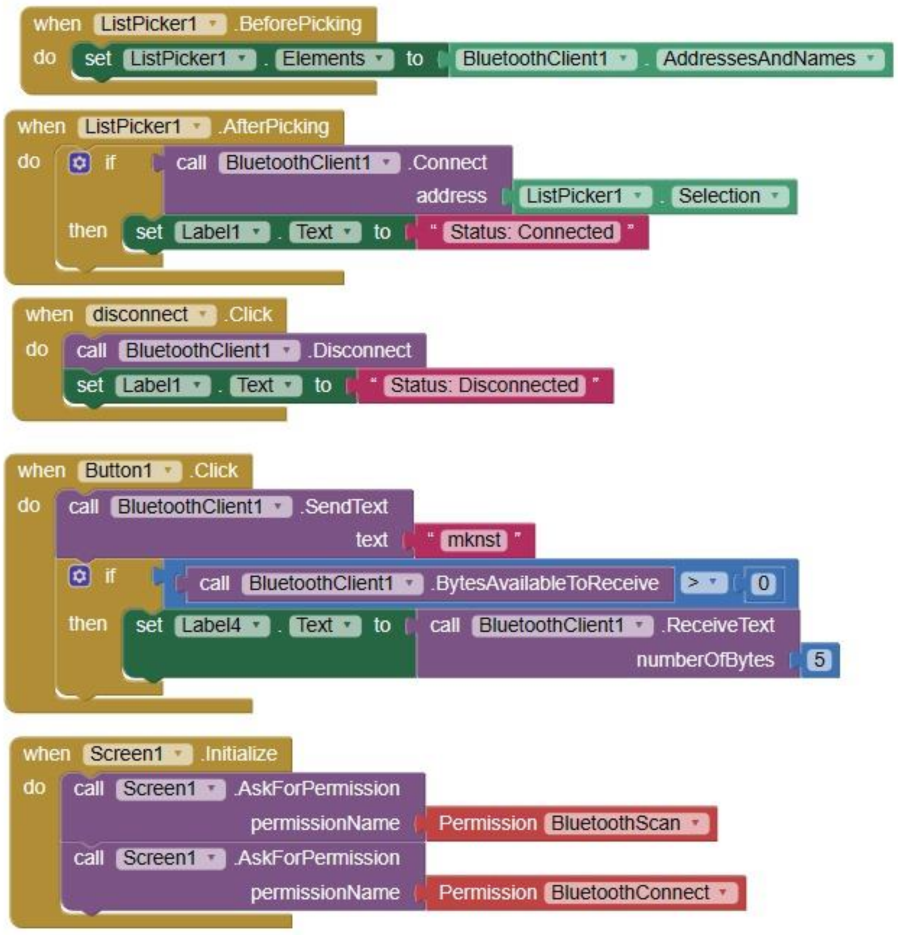


Status: Connected

Which Component to test?

Capcitance value: 220.00 nF

CODE of the application-



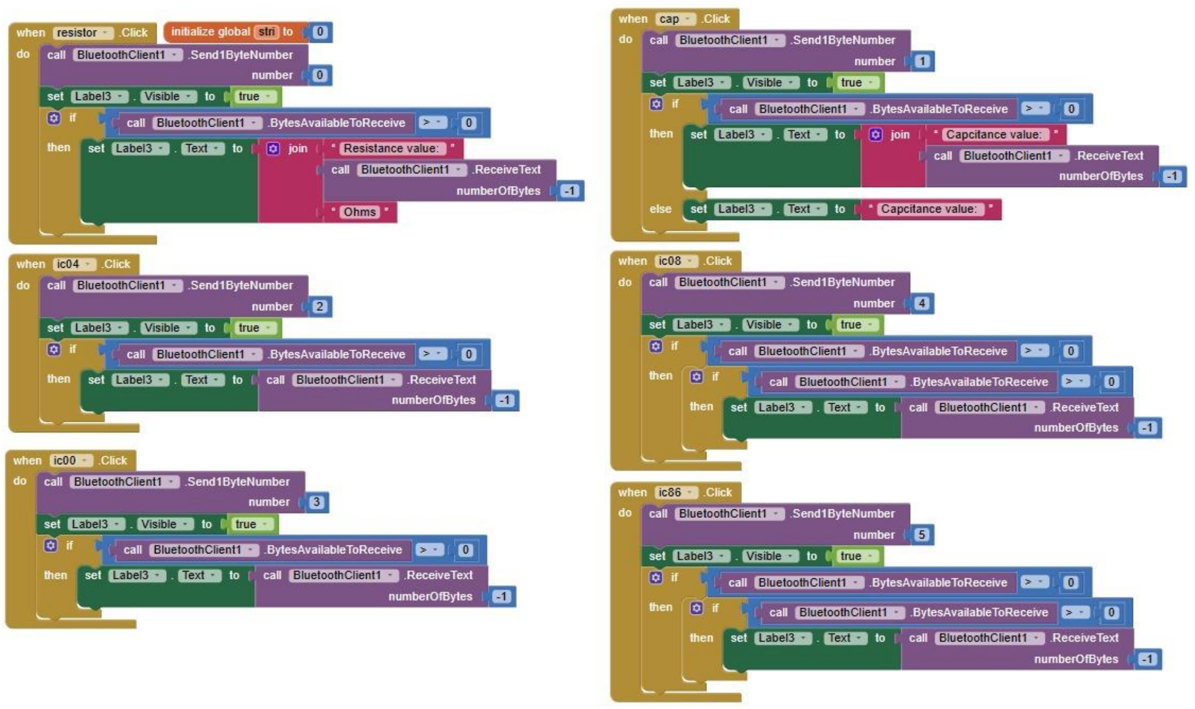
```
when ListPicker1 .BeforePicking
do  set ListPicker1 . Elements to  BluetoothClient1 . AddressesAndNames

when ListPicker1 .AfterPicking
do  if    call BluetoothClient1 .Connect
                              address  ListPicker1 . Selection
    then  set Label1 . Text to  " Status: Connected "

when disconnect .Click
do  call BluetoothClient1 .Disconnect
    set Label1 . Text to  " Status: Disconnected "

when Button1 .Click
do  call BluetoothClient1 .SendText
                     text  " mknst "
    if    call BluetoothClient1 .BytesAvailableToReceive  >  0
    then  set Label4 . Text to  call BluetoothClient1 .ReceiveText
                                           numberOfBytes  5

when Screen1 .Initialize
do  call Screen1 .AskForPermission
             permissionName  Permission BluetoothScan
    call Screen1 .AskForPermission
             permissionName  Permission BluetoothConnect
```



```
when resistor .Click                              initialize global stri to  0
do  call BluetoothClient1 .Send1ByteNumber
                                  number  0
    set Label3 . Visible to  true
    if    call BluetoothClient1 .BytesAvailableToReceive  >  0
    then  set Label3 . Text to  join  " Resistance value: "
                                      call BluetoothClient1 .ReceiveText
                                                     numberOfBytes  -1
                                      " Ohms "

when ic04 .Click
do  call BluetoothClient1 .Send1ByteNumber
                                  number  2
    set Label3 . Visible to  true
    if    call BluetoothClient1 .BytesAvailableToReceive  >  0
    then  set Label3 . Text to  call BluetoothClient1 .ReceiveText
                                            numberOfBytes  -1

when ic00 .Click
do  call BluetoothClient1 .Send1ByteNumber
                                  number  3
    set Label3 . Visible to  true
    if    call BluetoothClient1 .BytesAvailableToReceive  >  0
    then  set Label3 . Text to  call BluetoothClient1 .ReceiveText
                                            numberOfBytes  -1

when cap .Click
do  call BluetoothClient1 .Send1ByteNumber
                                  number  1
    set Label3 . Visible to  true
    if    call BluetoothClient1 .BytesAvailableToReceive  >  0
    then  set Label3 . Text to  join  " Capcitance value: "
                                      call BluetoothClient1 .ReceiveText
                                                     numberOfBytes  -1
    else  set Label3 . Text to  " Capcitance value: "

when ic08 .Click
do  call BluetoothClient1 .Send1ByteNumber
                                  number  4
    set Label3 . Visible to  true
    if    call BluetoothClient1 .BytesAvailableToReceive  >  0
    then  if    call BluetoothClient1 .BytesAvailableToReceive  >  0
          then  set Label3 . Text to  call BluetoothClient1 .ReceiveText
                                                  numberOfBytes  -1

when ic86 .Click
do  call BluetoothClient1 .Send1ByteNumber
                                  number  5
    set Label3 . Visible to  true
    if    call BluetoothClient1 .BytesAvailableToReceive  >  0
    then  if    call BluetoothClient1 .BytesAvailableToReceive  >  0
          then  set Label3 . Text to  call BluetoothClient1 .ReceiveText
                                                  numberOfBytes  -1
```

Future Improvements:

Expanding the range of capacitance and resistance measurements beyond the current range limits presents a significant opportunity for improvement in our project. By incorporating components and circuitry capable of handling a wider range of values, we can accommodate a broader spectrum of electronic devices and experimental setups, thereby enhancing the versatility and applicability of the component tester.
Additionally, increasing the number of supported ICs, currently set as 4 to a higher quantity would further broaden the tool's utility and relevance in diverse laboratory settings. By addressing these aspects, we can enhance the effectiveness and usability of the component tester.

Arduino Code

Code for Arduino1-

```
#include <Wire.h>

int i1 = 13;
int i2 = 12;
int i3 = 11;
int i4 = 10;
int i5 = 9;
int i6 = 8;
// i7 is ground
int i8 = 7;
int i9 = 6;
int i10 = 5;
int i11 = 4;
int i12 = 3;
int i13 = 2;
// i14 is vcc

int x = -1;
int result;

void setup() {

  Serial.begin(9600);

  Wire.begin(9);
  Wire.onReceive(receiveEvent);

}

void receiveEvent(int bytes){
  x = Wire.read();
}

bool gate7400(int input1, int input2, int output1){
  pinMode(input1, OUTPUT);
  pinMode(input2, OUTPUT);
  pinMode(output1, INPUT);

  digitalWrite(input1, LOW);
  digitalWrite(input2, LOW);
  if(digitalRead(output1) == LOW) return 0;

  digitalWrite(input1, HIGH);
  digitalWrite(input2, LOW);
  if(digitalRead(output1) == LOW) return 0;

  digitalWrite(input1, LOW);
  digitalWrite(input2, HIGH);
  if(digitalRead(output1) == LOW) return 0;

  digitalWrite(input1, HIGH);
  digitalWrite(input2, HIGH);
  if(digitalRead(output1) == HIGH) return 0;

  return 1;
}

bool ic7400(){
  if(gate7400(i1, i2, i3) == 0) return 0;
  if(gate7400(i4, i5, i6) == 0) return 0;
  if(gate7400(i10, i9, i8) == 0) return 0;
  if(gate7400(i13, i12, i11) == 0) return 0;

  return 1;
}

bool gate7408(int input1, int input2, int output1){
  pinMode(input1, OUTPUT);
  pinMode(input2, OUTPUT);
  pinMode(output1, INPUT);

  digitalWrite(input1, LOW);
  digitalWrite(input2, LOW);
  if(digitalRead(output1) == HIGH) return 0;

  digitalWrite(input1, HIGH);
  digitalWrite(input2, LOW);
  if(digitalRead(output1) == HIGH) return 0;

  digitalWrite(input1, LOW);
  digitalWrite(input2, HIGH);
  if(digitalRead(output1) == HIGH) return 0;

  digitalWrite(input1, HIGH);
  digitalWrite(input2, HIGH);
  if(digitalRead(output1) == LOW) return 0;

  return 1;
}

bool ic7408(){
  if(gate7408(i1, i2, i3) == 0) return 0;
  if(gate7408(i4, i5, i6) == 0) return 0;
  if(gate7408(i10, i9, i8) == 0) return 0;
  if(gate7408(i13, i12, i11) == 0) return 0;

  return 1;
}

bool gate7486(int input1, int input2, int output1){
  pinMode(input1, OUTPUT);
  pinMode(input2, OUTPUT);
  pinMode(output1, INPUT);

  digitalWrite(input1, LOW);
  digitalWrite(input2, LOW);
  if(digitalRead(output1) == HIGH) return 0;

  digitalWrite(input1, HIGH);
  digitalWrite(input2, LOW);
  if(digitalRead(output1) == LOW) return 0;

  digitalWrite(input1, LOW);
  digitalWrite(input2, HIGH);
  if(digitalRead(output1) == LOW) return 0;

  digitalWrite(input1, HIGH);
  digitalWrite(input2, HIGH);
  if(digitalRead(output1) == HIGH) return 0;

  return 1;
}

bool ic7486(){
  if(gate7486(i1, i2, i3) == 0) return 0;
  if(gate7486(i4, i5, i6) == 0) return 0;
  if(gate7486(i10, i9, i8) == 0) return 0;
  if(gate7486(i13, i12, i11) == 0) return 0;

  return 1;
}

bool gate7404(int input1, int output1){
  pinMode(input1, OUTPUT);
  pinMode(output1, INPUT);

  digitalWrite(input1, HIGH);
  if(digitalRead(output1) == HIGH) return 0;

  digitalWrite(input1, LOW);
  if(digitalRead(output1) == LOW) return 0;

  return 1;
}
```

```
bool ic7404(){
  if(gate7404(i1, i2) == 0) return 0;
  if(gate7404(i3, i4) == 0) return 0;
  if(gate7404(i5, i6) == 0) return 0;
  if(gate7404(i9, i8) == 0) return 0;
  if(gate7404(i11, i10) == 0) return 0;
  if(gate7404(i13, i12) == 0) return 0;

  return 1;
}

void loop() {

  if(x == 2 || x == 3 || x == 4 || x == 5){
    if(x == 2){
      result = ic7404();
    }
    else if(x == 3){
      result = ic7400();
    }
    else if(x == 4){
      result = ic7408();
    }
    else if(x == 5){
      result = ic7486();
    }
    pinMode(i1, OUTPUT);
    if(result) digitalWrite(i1, HIGH);
    else digitalWrite(i1, LOW);
    if(result) Serial.println("Working good");
    else Serial.println("Not working");
    Serial.println(x);
    // delay(4000);
    x = -1;
  }

}
```

```
    // Add code here to
```

Code for Arduino2-

```
#include <Wire.h>
#include <SoftwareSerial.h>
SoftwareSerial BTserial(2, 3);  // RX | TX

int received = 5;

int sensorPin = 7;
int Vin = 5;
float Vout = 0;
float R1 = 1000;
float R2 = 0;
float buffer = 0;
int raw = 0;

int chargePin = 12;
int dischargePin = 11;
int analogPin = 2;
float Rc = 100000;
unsigned long startTime;
unsigned long elapsedTime;
float microFarads;
float nanoFarads;

void setup() {
  BTserial.begin(9600);
  Serial.begin(9600);
  pinMode(chargePin, OUTPUT);
  digitalWrite(chargePin, LOW);

  Wire.begin();
}

void resistance(){
  // Resistance
  raw = analogRead(sensorPin);
  if (raw) {
    buffer = raw * Vin;
    Vout = (buffer) / 1024.0;
    buffer = (Vin / Vout) - 1;
    R2 = R1 * buffer;
    BTserial.println((String)R2);
  }
}

void capacitance(){
  //Capacitance
  digitalWrite(chargePin, HIGH);  // set chargePin HIGH and capacitor charging
  startTime = millis();
  while(analogRead(analogPin) < 648){}
  elapsedTime= millis() - startTime;
  microFarads = ((float)elapsedTime / Rc) * 1000;
  if (microFarads > 1){
    BTserial.println((String)microFarads + " µF");
  }
  else
  {
    microFarads = ((float)elapsedTime * 1e6) / Rc;
    nanoFarads = microFarads ;
    BTserial.println((String)nanoFarads + " nF");
  }
  digitalWrite(chargePin, LOW);
  pinMode(dischargePin, OUTPUT);
  digitalWrite(dischargePin, LOW);
  while(analogRead(analogPin) > 0){}
  pinMode(dischargePin, INPUT);
}

void loop() {

  if(BTserial.available()){
    int s = BTserial.read();

    if(s == 0){
      resistance();
    }
    else if(s == 1){
      capacitance();
    }
    else if(s == 2 || s == 3 || s == 4 || s == 5){
      Wire.beginTransmission(9);
      Wire.write(s);
      Wire.endTransmission();
      delay(300);
      int ans = digitalRead(received);
      if(s == 2){
        if(ans) BTserial.println("IC 7404 is working fine");
        else BTserial.println("IC 7404 is not working fine");
      }
      else if(s == 3){
        if(ans) BTserial.println("IC 7400 is working fine");
        else BTserial.println("IC 7400 is not working fine");
      }
      else if(s == 4){
        if(ans) BTserial.println("IC 7408 is working fine");
        else BTserial.println("IC 7408 is not working fine");
      }
      else if(s == 5){
        if(ans) BTserial.println("IC 7486 is working fine");
        else BTserial.println("IC 7486 is not working fine");
      }
      Serial.println(ans);
    }
  }
}
```