# School of Computer Science and Engineering

**CSE-2006-Microprocesor and Interfacing**

**Winter Semester 2020-21**

**Slot – D1**

# REPORT

**Under the guidance of**
Prof. Florence Gnana Poovathy J
Assistant Professor Senior
SENSE

**TEAM MEMBERS:**
Mihir Antwal (19BCE1641)
Kaustubh P Gholap (19BCE1684)
Sam Methuselah (19BCE1698)
Arjun Arora (19BCE1808)

# Honeypot Server

# DECLARATION

We hereby declare that the project entitled "Honeypot Server" submitted by 19BCE1641-Mihir Antwal, 19BCE1684-Kaustubh P Gholap, 19BCE1698-Sam Methuselah and 19BCE1808-Arjun Arora, for the project component of the Microprocessor and Interfacing to VIT is a record of bonafide carried out by us under the supervision of Prof. Florence Gnana Poovathy J, Assistant Professor Senior, SENSE, Vellore Institute of Technology, Chennai.

We further declare that the project report submitted has not been submitted and will not be submitted, either in a part or full for the award of any other degree or diploma in this institute or any other Institute or University.

**PLACE:** CHENNAI

**SIGNATURE:**

[illegible] Mihir [illegible] Arjun

**DATE:** 25 MAY, 2021

# CERTIFICATE

This is to certify that the project entitled “**Honeypot Server**”, submitted by Student Name and Register Number for the project component of the Microprocessor and Interfacing to VIT is a record of bona-fide work carried out by them under my supervision during the period, 01.02.2021 to 08.06.2020 as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The project fulfils the requirements and regulations of the university and in my opinion meets the necessary standards for submission.

PLACE: CHENNAI Signature of the guide

DATE: 25 MAY, 2021 Prof. Florence Gnana Poovathy J
Assistant Professor Senior
SENSE

## ABSTRACT

Honeypots are usually deployed inside production networks alongside production servers, the honeypot plays the role of a decoy as part of the production network intrusion detection system (IDS). A production honeypot is designed to appear real and contains duplicate information to attract and occupy hackers to tie up their time and resources, ultimately giving administrators time to assess and mitigate any vulnerabilities in their actual production systems.

A honeypot server consists of a computer, applications and data that simulate the behaviour of a real system and appears as part of a network; however, the honeypot is actually isolated. Viewing and logging this activity can help improve security by providing insight into the level and types of threat a network infrastructure faces while distracting attackers away from assets of real value. Researchers suspect that some cybercriminals use honeypots themselves to gather intelligence about researchers, act as decoys and to spread misinformation.

The aim of the project is to make a honeypot using Arduino. It is cost efficient because they do not require high-performance resources to process large volumes of network traffic looking for attacks, because they only interact with malicious activities.

So, it can be used by small companies as they collect data from actual attacks and other unauthorized activities, providing analysts with a rich source of useful information and it is very Cost efficient.

## ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Prof. Florence Gnana Poovathy J**, Assistant Professor Senior, SENSE, for her consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We are extremely grateful to **Dr. Sivasubramanian. A**, Dean of School of Electronics Engineering, VIT Chennai, for extending the facilities of the School towards our project and for his unstinting support.

We express our thanks to our Head of the Department **Dr. Vetrivelan. P** for his support throughout the course of this project.

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the course.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

| | | | |
|---|---|---|---|
| Mihir | Sai | Arjun | [illegible] |
| **Mihir Antwal** | **Kaustubh P Gholap** | **Arjun Arora** | **Sam Methuselah** |
| (19BCE1641) | (19BCE1684) | (19BCE1808) | (19BCE1698) |

# TABLE OF CONTENTS

# INTRODUCTION

A honeypot is a network-attached system set up as a decoy to lure cyber attackers and to detect, deflect or study hacking attempts in order to gain unauthorized access to information systems. The function of a honeypot is to represent itself on the internet as a potential target for attackers usually a server or other high-value target and to gather information and notify defenders of any attempts to access the honeypot by unauthorized users.

Honeypot systems often use hardened operating systems and are usually configured so that they appear to offer attackers exploitable vulnerabilities. Honeypots are most often used by large enterprises and by companies involved in cyber security research, to identify and defend attacks from advanced persistent threat actors. Honeypots can be an important tool for large organizations to take an active defense stance against attackers, or for cyber security researchers who want to learn more about the tools and techniques that attackers use.

**Honeypots** are run to gather information about the motives and tactics of the black hat community targeting different networks. These honeypots do not add direct value to a specific organization; instead, they are used to research the threats that organizations face and to learn how to better protect against those threats. Research honeypots are complex to deploy and maintain, capture extensive information, and are used primarily by research, military, or government organizations.

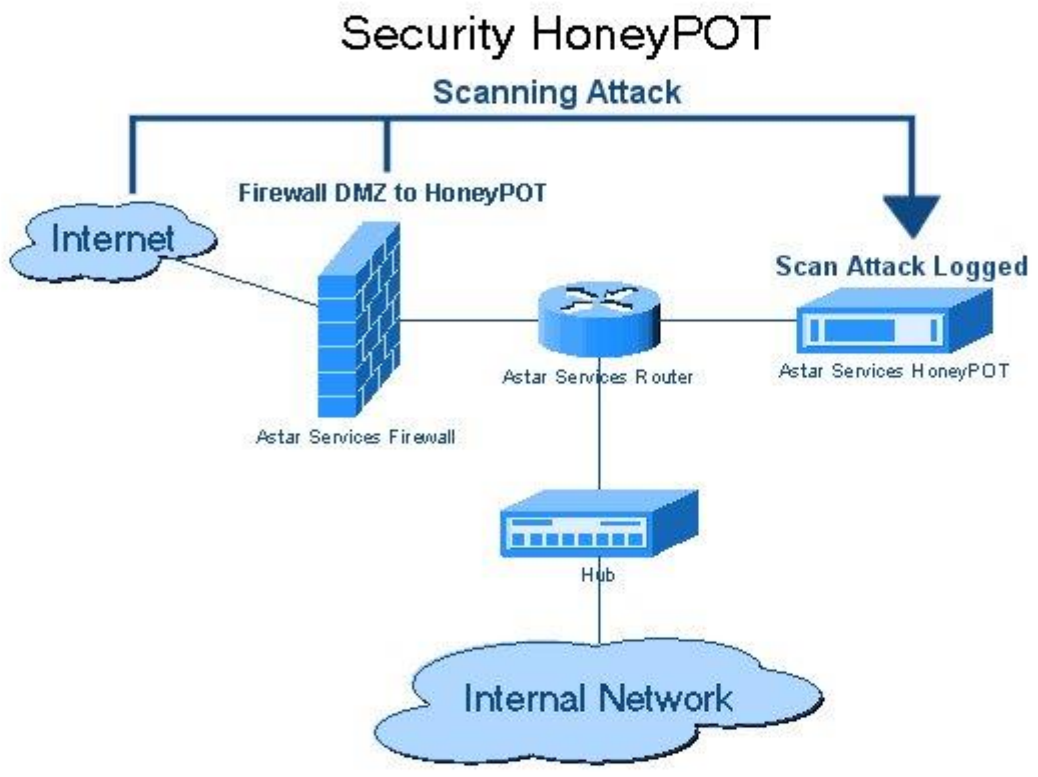
**Based on design criteria, honeypots can be classified as:**

- **Pure honeypots**
- **High-Interaction honeypots**
- **Low-Interaction honeypots**

**Pure honeypots** are full-fledged production systems. The activities of the attacker are monitored by using a bug tap that has been installed on the honeypot's link to the network. No other software needs to be installed. Even though a pure honeypot is useful, stealthiness of the defense mechanisms can be ensured by a more controlled mechanism.

**High-interaction honeypots** imitate the activities of the production systems that host a variety of services and, therefore, an attacker may be allowed a lot of services to waste their time. By employing virtual machines, multiple honeypots can be hosted on a single physical machine. Therefore, even if the honeypot is compromised, it can be restored more quickly. In general, high-interaction honeypots provide more security by being difficult to detect, but they are expensive to maintain. If virtual machines are not available, one physical computer must be maintained for each honeypot, which can be exorbitantly expensive. Example: Honeynet.

**Low-interaction honeypots** simulate only the services frequently requested by attackers. Since they consume relatively few resources, multiple virtual machines can easily be hosted on one physical system, the virtual systems have a short response time, and less code is required, reducing the complexity of the virtual system's security.



## PROJECT SCOPE-

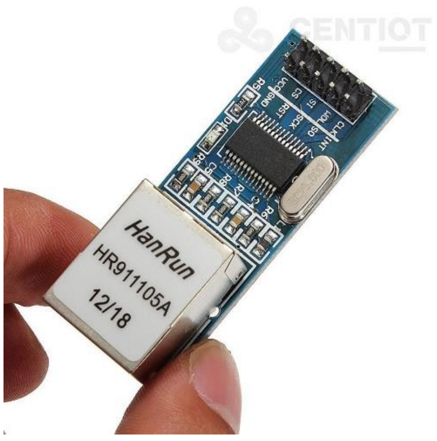
- **Collect real data:** Honeypots collect data from actual attacks and other unauthorized activities, providing analysts with a rich source of useful information.
- **Reduce false positives**: Ordinary cybersecurity detection technologies generate alerts that can include a significant volume of false positives, but honeypots reduce this volume because there is no reason for legitimate users to access them.
- **Cost-effective:** Honeypots can be good investments because they do not require high-performance resources to process large volumes of network traffic looking for attacks, because they only interact with malicious activities.
- **Encryption:** Honeypots capture malicious activity, even if an attacker is using encryption.

# PROJECT RESOURCE REQUIREMENTS

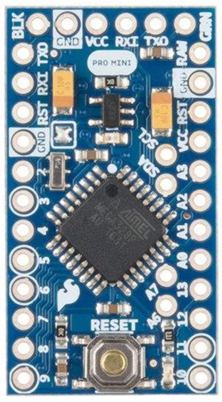
## COMPONENTS REQUIRED -

- 128×64 Blue OLED display-

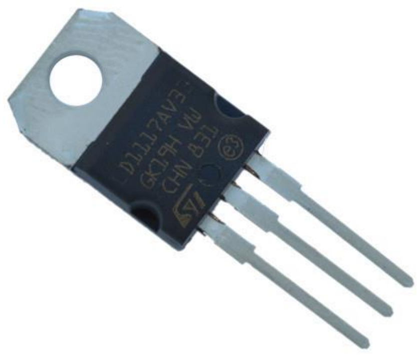
- ENC28J60 Ethernet module-



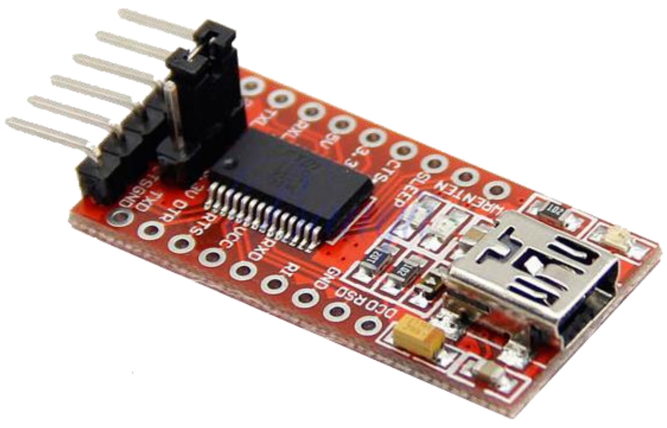
- Arduino Pro Mini 3.3v 8mhz-



- LM1117 3.3 volt voltage regulator-



- FTD1232 module-



- 2x10K resistors-

- 22uF capacitor-

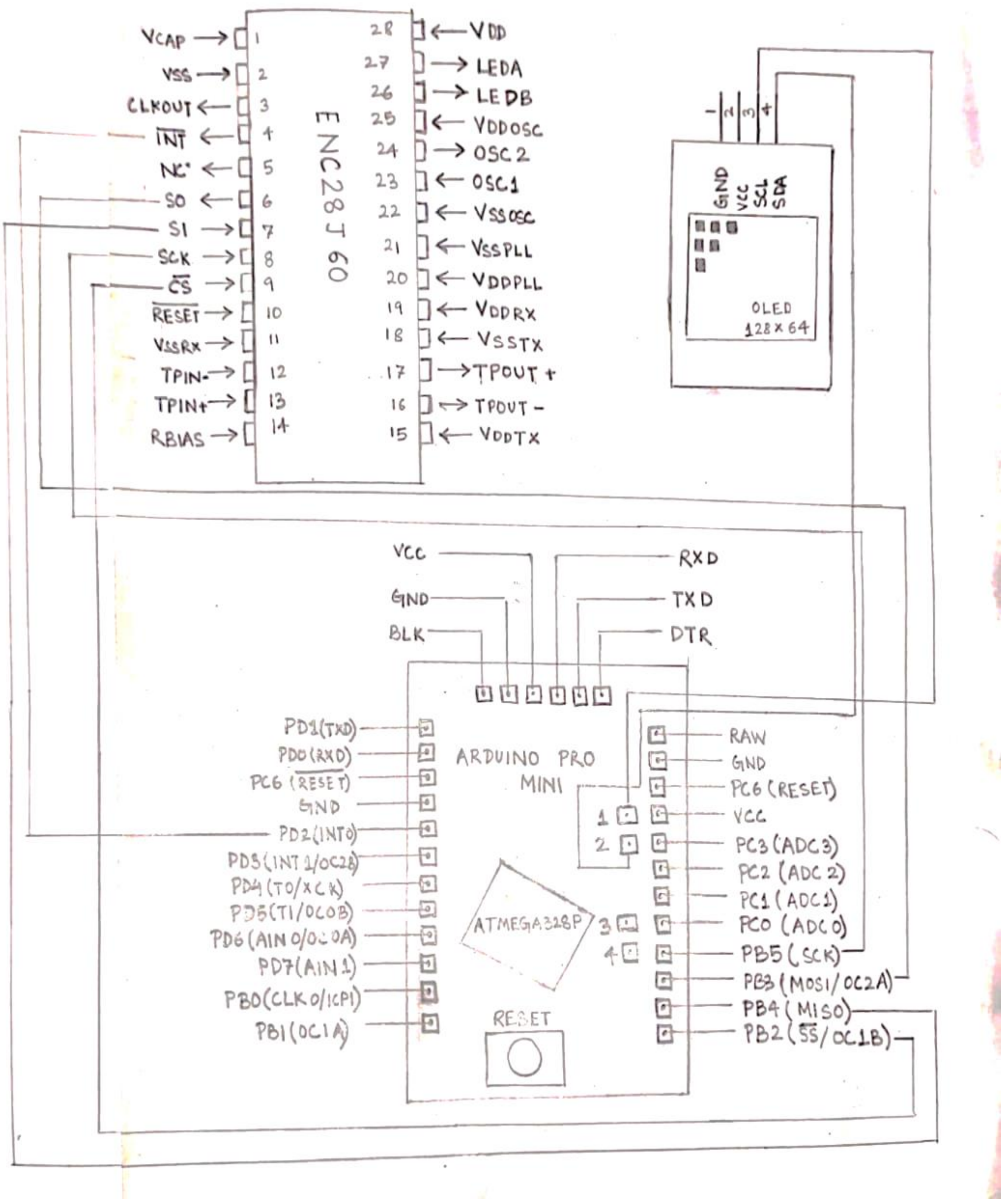
## SOFTWARE REQUIREMENTS-

- Arduino IDE
- OrCAD
- Proteus

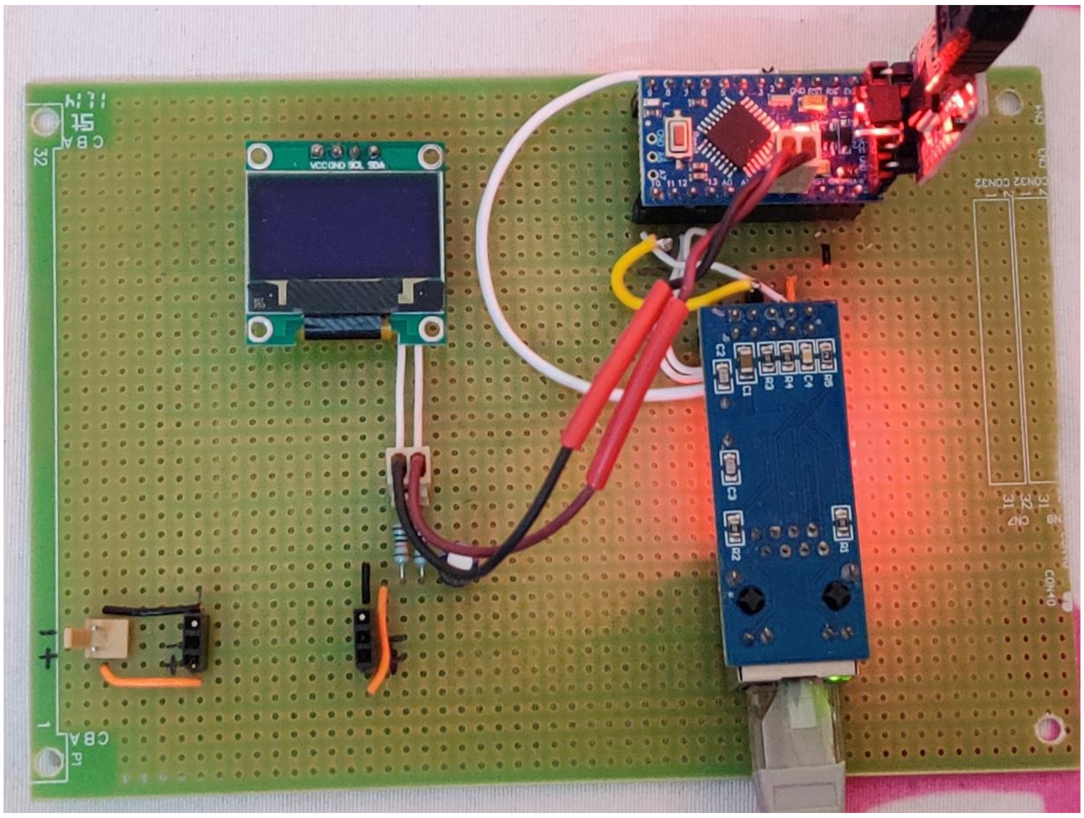
## SYSTEM REQUIREMENTS-

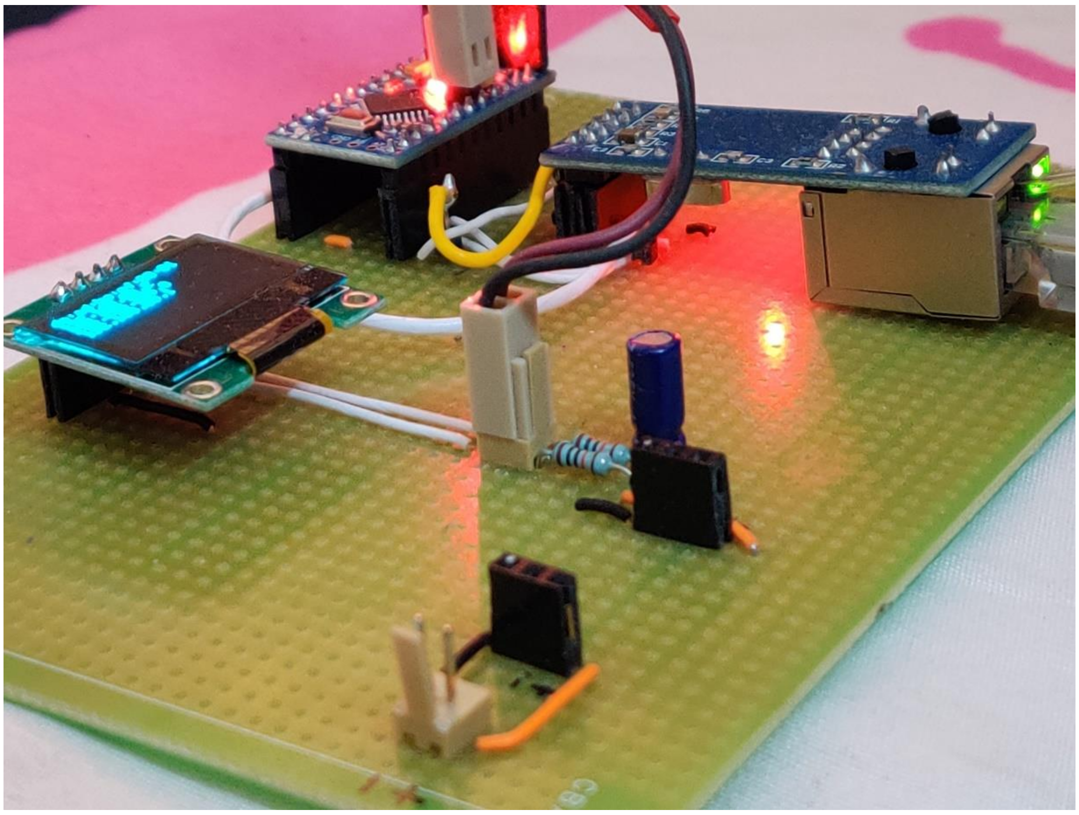
- Ram: min 8GB
- Storage: 20GB and above
- Monitor: 1920*1080
- Processor: Intel or AMD 2GHz and above
- OS: Windows 7 or higher

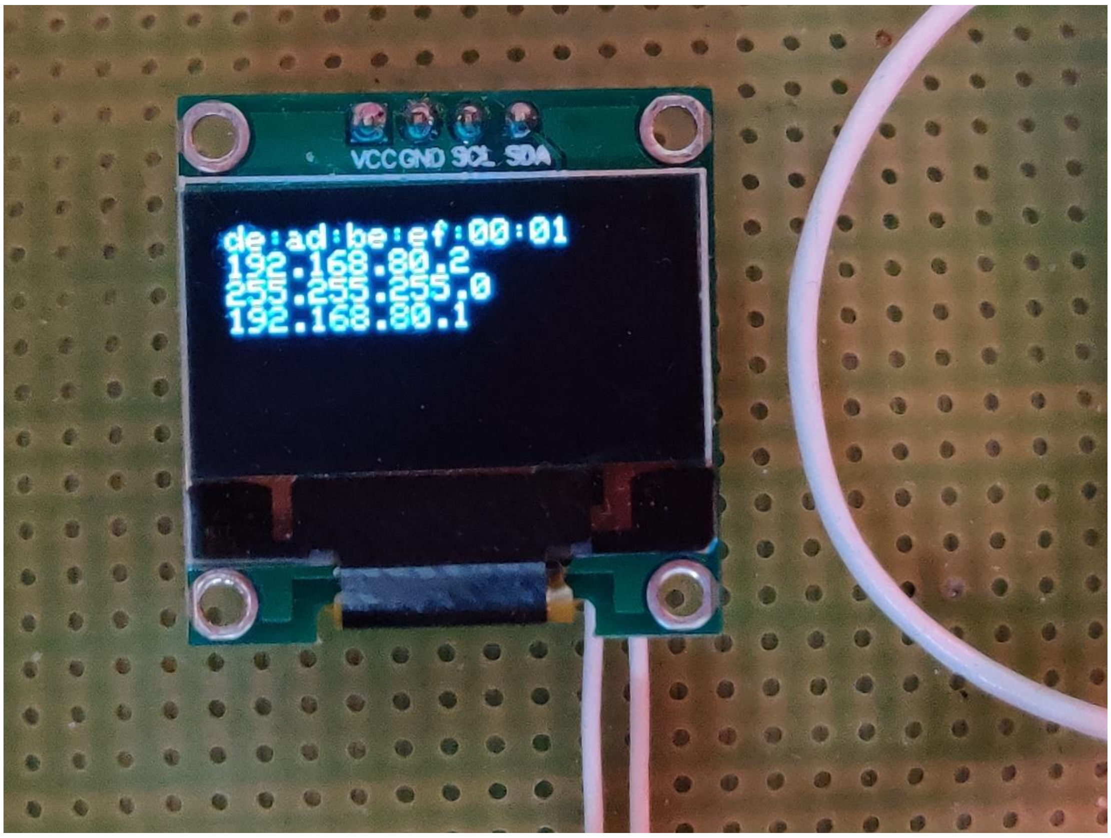
## CIRCUIT DIAGRAM



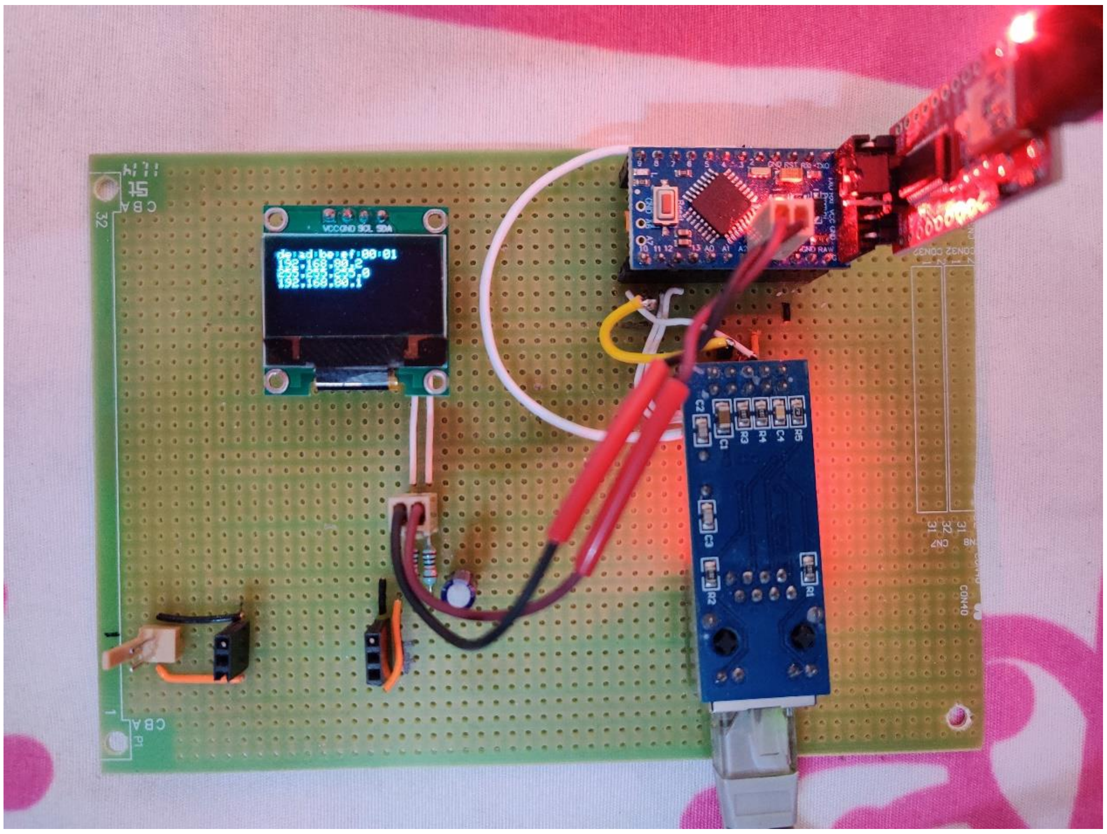
# IMPLEMENTATION







```
de:ad:be:ef:00:01
192.168.80.2
255.255.255.0
192.168.80.1
```



```
de:ad:be:ef:00:01
192.168.80.2
255.255.255.0
192.168.80.1
```

The project aims to make a honeypot server using Arduino which will detect incoming packets to the server and display its IP address on the OLED screen.

The working-

- When powered up it displays it's address (can be obtained via DHCP or set statically in the code).
- After 20 seconds it blanks the display and listens for incoming traffic to the device.
- If it receives an ICMP, a UDP or a TCP SYN packet it shows up on the display.
- As, nothing in the native network should be scanning or connecting to this IP. If this display shows something, it means we got intruders poking around in the network.

## Advantages of Honeypots

Honeypots only collect data when someone or something is interacting with them. Organizations that may log thousands of alerts a day may only log a hundred alerts with honeypots. This makes the data honeypots collect much easier to manage and analyze.

- **Reduced False Positives:**

  Honeypots dramatically reduce false positives. Any activity with honeypots is by definition unauthorized, making it extremely effective at detecting attacks. This allows organizations to quickly and easily reduce, if not eliminate, false alerts, allowing organizations to focus on other security priorities, such as patching.

- **Catching False Negatives**

  Honeypots can easily identify and capture new attacks or actions against them. Any activity with the honeypot is an anomaly, making new or unseen attacks easily stand out.

- **Minimal Resources**

  Honeypots require minimal resources, even on the largest of networks. A simple Pentium computer can monitor literally millions of IP addresses on an OC-12 network.

- **Encryption**

  It does not matter if an attack is encrypted, the honeypot will capture the activity.

- **Protocol Independent**

  It does not matter which IP protocol an attacker uses, honeypots will detect, capture, and log all IP activity. In one documented case, a Solaris honeypot detected and captured an attack where attackers attempted to hide their communications using IPv6 tunnelling within IPv4. On the other hand, there are almost no NIDS (Network intrusion detection system) technologies that can decode IPv6 or IPv6-tunneled traffic.

- **Intelligence Gathering**

  Honeypots can gather a lot of valuable information about the attackers, and also the nature of their attacks, which can be used to take appropriate action against them. A honeypot is a valuable resource, especially to collect information about the proceedings of attackers as well as their deployed tools.

## Disadvantages Of Honeypots

With all of these wonderful advantages, you would think honeypots would be the ultimate security solution. Unfortunately, that is not the case. They have several disadvantages. It is because of these disadvantages that honeypots do not replace any security mechanisms; they only work with and enhance your overall security architecture.

- **Narrow Field Of View**

  The greatest disadvantage of honeypots is they have a narrow field of view: They only see what activity is directed against them. If an attacker breaks into your network and attacks a variety of systems, your honeypot will be blissfully unaware of the activity unless it is attacked directly. If the attacker has identified your honeypot for what it is, she can now avoid that system and infiltrate your organization, with the honeypot never knowing she got in. As noted earlier, honeypots have a microscope effect on the value of the data you collect, enabling you to focus closely on data of known value. However, like a microscope, the honeypot's very limited field of view can exclude events happening all around it.

- **Finger Printing**

  Another disadvantage of honeypots, especially many commercial versions, is fingerprinting. Fingerprinting is when an attacker can identify the true identity of a honeypot because it has certain expected characteristics or behaviours. For example, a honeypot may emulate a Web server. Whenever an attacker connects to this specific type of honeypot, the Web server responds by sending a common error message using standard HTML. This is the exact response we would expect for any Web server. However, the honeypot has a mistake in it and misspells one of the HTML commands, such as spelling the word length as legnht. This misspelling now becomes a fingerprint for the honeypot, since any attacker can quickly identify it because of this error in the Web server emulation. An incorrectly implemented honeypot can also identify itself. For example, a honeypot may be designed to emulate an NT IIS Web server, but the honeypot also has certain characteristics that identify it as a Unix Solaris server. These contradictory identities can act as a signature for a honeypot

If a blackhat identifies an organization using a honeypot on its internal networks, he could spoof the identity of other production systems and attack the honeypot. The honeypot would detect these spoofed attacks, and falsely alert administrators that a production system was attacking it, sending the organization on a wild goose chase. Meanwhile, in the midst of all the confusion, an attacker could focus on real attacks.

Fingerprinting is an even greater risk for research honeypots. A system designed to gain intelligence can be devastated if detected. An attacker can feed bad information to a research honeypot as opposed to avoiding detection. This bad information would then lead the security community to make incorrect conclusions about the blackhat community.

This is not to say all honeypots must avoid detection. Some organizations might want to scare away or confuse attackers. Once a honeypot is attacked, it can identify itself and then warn off the attacker in hopes of scaring him off. However, in most situations organizations do not want honeypots to be detected.

- **Risk**

The third disadvantage of honeypots is risk: They can introduce risk to your environment. By risk, we mean that a honeypot, once attacked, can be used to attack, infiltrate, or harm other systems or organizations. As we discuss later, different honeypots have different levels of risk. Some introduce very little risk, while others give the attacker entire platforms from which to launch new attacks. The simpler the honeypot, the less the risk. For example, a honeypot that merely emulates a few services is difficult to compromise and use to attack other systems. In contrast, a honeypot that creates a jail gives an attacker an actual operating system with which to interact. An attacker might be able to break out of such a cage and then use the honeypot to launch passive or active attacks against other systems or organizations. Risk is variable, depending on how one builds and deploys the honeypot.

Because of their disadvantages, honeypots cannot replace other security mechanisms such as firewalls and intrusion detection systems. Rather, they add value by working with existing security mechanisms. They play a part in your overall defense.

# CODE-

## Honeypot.ino-

```
#include <Arduino.h>

#include "ScrollDisplay.h"

#include "Listener.h"

#include "EtherCard.h"

#include "Wire.h"



// Here you setup static IP or DHCP

static bool dhcp = true;

static byte myip[] = { 192,168,80,2 }; // Only used if you want static IP

static byte gwip[] = { 192,168,80,1 }; // Only used if you want static IP

static byte mymac[] = { 0xDE,0xAD,0xBE,0xEF,0x00,0x01 }; // Set this to something clever.



// Global objects

ListenerClass listener;

ScrollDisplay disp( 4 );



void setup()

{

  Wire.begin();
  Serial.begin( 115200 );
  if ( dhcp ) listener.connect( mymac );
  if (!dhcp)  listener.connect(myip,gwip,mymac);
  disp.setup( 0x3C );
  showInfo();
}


void loop() {
  static char msg[20];
  if ( listener.gotPacket() ) {
    if ( listener.isForMe() ) {
      if ( listener.isUdp() ) sprintf( msg, "UDP port %d src=", listener.getDstPort() );
      if ( listener.isTcpSyn() )sprintf( msg, "TCP port %d src=", listener.getDstPort() );
      if ( listener.isIcmpPingReq() ) sprintf( msg, "ICMP src=" );
      disp.printScrollLine( msg );

      listener.getSrcIp( msg );
      disp.printScrollLine( msg );
    }
  }
  disp.handle();
}
```

```
void showInfo() {
  char info[20];
  listener.getMyMac( info );
  disp.printScrollLine( info );
  listener.getStrIp( info, ether.myip );
  disp.printScrollLine( info );
  listener.getStrIp( info, ether.netmask );
  disp.printScrollLine( info );
  listener.getStrIp( info, ether.gwip );
  disp.printScrollLine( info );

  disp.blankAfter( 20 );
}
```

## **Listener.cpp-**

```
#include "Listener.h"
#include "EtherCard.h"



byte Ethernet::buffer[500];



// Connectes to the enc28j60 network module with a static IP address
void ListenerClass::connect(byte* ip, byte* gw, byte* mac) {
        memcpy( this->ip, ip, 4 );
        memcpy( this->gw, gw, 4 );
        memcpy( this->mac, mac, 6 );

        Serial.println( "Connecting to ethernet controller 1" );
        while ( ether.begin( sizeof Ethernet::buffer, this->mac ) == 0 ) {
                Serial.println( "Failed to access Ethernet controller" );
                delay( 1000 );
        }

        Serial.println( "Connected" );
        ether.staticSetup( this->ip, this->gw );
}


// Connectes to the enc28j60 network module and gets DHCP address
void ListenerClass::connect( byte* mac ) {
        memcpy( this->mac, mac, 6 );

        Serial.println( "Connecting to ethernet controller" );
        while ( ether.begin( sizeof Ethernet::buffer, this->mac ) == 0 ) {
                Serial.println( "Failed to access Ethernet controller" );
                delay( 1000 );
        }

        Serial.println( "Connected" );
```

```
    if ( !ether.dhcpSetup() )
        Serial.println( F( "DHCP failed" ) );
}



// Returns true if a packet was received
bool ListenerClass::gotPacket() {
    ether.packetLoop( packetLength );

    packetLength = ether.packetReceive();
    return packetLength > 0;

}



// Returns true if the last received packet was for the IP of the device
bool ListenerClass::isForMe() {
    for ( uint8_t pos = 0; pos < 4; pos++ ) {
        if ( ether.myip[pos] != *( (byte*) Ethernet::buffer + 0x1e + pos ) ) {
            return false;
        }
    }
    return true;
}
```

```
// Returns true if the last received packet was an ICMP ping request

bool ListenerClass::isIcmpPingReq() {

        if ( getProtocol() == 1 ) {

                byte *type = (byte*) Ethernet::buffer + 0x22;

                if ( *type == 8 ) return true;

        }

        return false;

}



// Returns true if the last received packet was an UDP packet

bool ListenerClass::isUdp() {

        if ( getProtocol() == 17 ) {

                return true;

        }

        return false;

}



// Returns true if the last received packet was a TCP SYN/connect packet

bool ListenerClass::isTcpSyn() {


        if ( getProtocol() == 6 ) {

                byte *flags = (byte*) Ethernet::buffer + 0x2f;
                bool syn = *flags & 0b00000010;
                if ( syn ) {
                        return true;
                }
        }
        return false;
}



// Returns the protocol number of the last received packet (eg. 6=tcp, 17=udp etc)
uint8_t ListenerClass::getProtocol() {
        byte *protocol = (byte*) Ethernet::buffer + 0x17;
        return *protocol;
}



// Returns the destination port number that the last received packet was destined for
uint16_t ListenerClass::getDstPort() {
        uint8_t port_msb = *( Ethernet::buffer + 0x24 );
        uint8_t port_lsb = *( Ethernet::buffer + 0x25 );
        uint16_t dstPort = (uint16_t)port_msb * 256 + (uint16_t)port_lsb ;
        return dstPort;
}
```

```
// Returns a string with the source IP number of the last received packet
void ListenerClass::getSrcIp(char* str) {
        uint8_t b1 = *( (byte*) Ethernet::buffer + 0x1a );
        uint8_t b2 = *( (byte*) Ethernet::buffer + 0x1b );
        uint8_t b3 = *( (byte*) Ethernet::buffer + 0x1c );
        uint8_t b4 = *( (byte*) Ethernet::buffer + 0x1d );
        sprintf( str, "%d.%d.%d.%d", b1, b2, b3, b4 );
}


// Returns a string representation of an IP number in the ether library (byte array
representation)
void ListenerClass::getStrIp( char* str, uint8_t* ip ) {
        sprintf( str, "%d.%d.%d.%d", ip[0], ip[1], ip[2], ip[3] );
}


// Returns a string with the mac address of the device
void ListenerClass::getMyMac( char* str ) {
        sprintf( str, "%02x:%02x:%02x:%02x:%02x:%02x", ether.mymac[0],
ether.mymac[1], ether.mymac[2], ether.mymac[3], ether.mymac[4], ether.mymac[5] );
}


// Prints to serial a hex representation of the last received packet
```

```
void ListenerClass::debug() {
        unsigned char *var = (byte*) Ethernet::buffer;
        String hexString;
        for ( uint8_t i = 0; i < packetLength; i++ ) { // traverse the var byte by byte
                char* p = (char *) var;

                uint8_t currentByte = *( p + i ); // get byte number i
                char currentByteHex[3];
                sprintf( currentByteHex, "%02X", currentByte ); // convert it to hex
                hexString = hexString + currentByteHex; // and concatenate it into a printable
string of all bytes

                if ( i != packetLength - 1 ) hexString = hexString + ",";
        }
        Serial.print( "Packet: " );
        Serial.println( hexString );
}
```

## Listener.h –

```
#ifndef _NETWORK_h
#define _NETWORK_h

#include <Arduino.h>

class ListenerClass {
```

```
protected:
        byte ip[4];
        byte gw[4];
        byte mac[6];
        int packetLength = 0;

public:
        void connect( byte * mac );
        void connect( byte * ip, byte * gw, byte * mac );
        void debug();

        bool gotPacket();
        bool isForMe();
        bool isIcmpPingReq();
        bool isUdp();
        bool isTcpSyn();
        uint8_t getProtocol();
        uint16_t getDstPort();
        void getSrcIp(char* str);
        void getStrIp( char * str, uint8_t * ip );
        void getMyMac( char * str );
};

#endif
```

## ScrollDisplay.cpp-

```
#include "ScrollDisplay.h"



// Initialises the display and clears it
void ScrollDisplay::setup( unsigned char displayAddress )
{
	line1[0] = line2[0] = line3[0] = line4[0] = 0;

	begin( SSD1306_SWITCHCAPVCC, displayAddress);
	clearDisplay();
	display();
}



// Should be called constantly from main loop
void ScrollDisplay::handle() {
	if ( blanking && millis() - blankStarted > (long) blanktime * 1000 )
	{
		blanking = false;
		char *q = (char*)"";
		for ( int i = 0; i < 4; i++ ) printScrollLine( q );
	}
}
```

```
// This sets the time that the display will be dimmed
void ScrollDisplay::blankAfter( uint16_t blankTime ) {
        blankStarted = millis();
        this->blanktime = blankTime;
        blanking = true;
}

// Scrolls up the existing lines in display and prints a new line in the buttom.
void ScrollDisplay::printScrollLine( char* line )
{
        scrollLinesUp();
        strncpy( line4, line, 19 );

        clearDisplay();
        setTextSize( 1 );
        setTextWrap( true );
        setTextColor( WHITE );
        setCursor( 0, 0 );

        println( line1 );
        println( line2 );
        println( line3 );
        println( line4 );
        display();
}



// Rotates all lines one position up.
void ScrollDisplay::scrollLinesUp()
{
        strncpy( line1, line2, 19 );
        strncpy( line2, line3, 19 );
        strncpy( line3, line4, 19 );
}
```

## ScrollDisplay.h-

```
#ifndef _SCROLLDISPLAY_h
#define _SCROLLDISPLAY_h

#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>



class ScrollDisplay : public Adafruit_SSD1306
{

protected:
        char line1[20];
        char line2[20];
        char line3[20];

        char line4[20];

        void scrollLinesUp();

        uint16_t blanktime;

        long blankStarted;

        bool blanking = false;


 public:

        ScrollDisplay( uint8_t x ) : Adafruit_SSD1306( x ) {};

        void setup( unsigned char displayAddress );

        void handle();

        void printScrollLine( char* line );

        void blankAfter( uint16_t blanktime );

};



#endif
```

# CONCLUSION

Honeypots are a highly flexible technology that can be applied to a variety of situations. As security tools, they have specific advantages. Specifically, honeypots collect small amounts of data, but most of this is information of high value. They have the ability to effectively work in resource intensive environments, and conceptually they are very simple devices. Also, they quickly demonstrate their value by detecting and capturing unauthorized activity.

However, honeypots share several major disadvantages. The most critical is they have a narrow field of view. If they are not attacked, they have no value. Second, certain honeypots can be fingerprinted, making detection possible. The third disadvantage is that honeypots can add additional risk: The honeypot may be used to attack or harm other systems or organizations. Any time you add additional services or applications to your environment, there are more things that can go wrong.

Within the three areas of security—prevention, detection and response—the primary value of production honeypots is detection. Because production honeypots greatly reduce the problem of both false negatives and false positives, they make an extremely efficient technology for detecting unauthorized activity. They also have some value with respect to reaction and, relatedly, helping organizations to develop their incident response skills. For prevention purposes, production honeypots are of minimal value. The concepts of deception and deterrence can be applied with honeypots to prevent attacks, but most organizations are better off spending their limited resources on security best practices, such as patching vulnerable services. Honeypots will not stop vulnerable systems from being hacked.

Research honeypots do not mitigate risk, but they primarily are used to gain information about threats. This information is then used to better understand and protect against these threats. When deploying honeypots, it is critical that organizations have a clearly defined security policy stating what activity is and is not authorized, including the use of honeypots to detect and monitor.

Honeypots are a new field in the sector of network security. Currently, there is a lot of ongoing research and discussions all around the world. No other mechanism is comparable in the efficiency of a honeypot if gathering information is a primary goal, especially if the tools an attacker uses are of interest. We are only beginning to see the potential of honeypots.

## REFERENCES:

1) https://en.wikipedia.org/wiki/Honeypot_(computing)

2) https://www.cse.wustl.edu/~jain/cse571-09/ftp/honey/

3) https://www.diva-portal.org/smash/get/diva2:327476/fulltext01

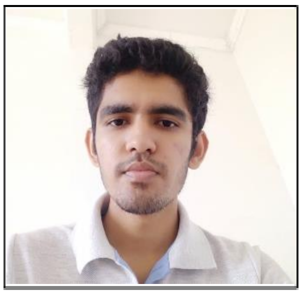
4) https://www.futurelearn.com/info/courses/security-operations/0/steps/89278

5) https://www.udemy.com/tutorial/mta-98-367-security-fundamentals-class-exam-prep-bundle/honeypots/

6) **Book-** Honeypots: Tracking Hackers

# BIODATA

| | |
|---|---|
| Name : | Mihir Antwal |
| Mobile Number : | 6003501683 |
| E-mail : | mihir.antwal2019@vitstudent.ac.in |
| Permanent Address: | D+ 30, Oil Housing Colony, Duliajan, Assam, Pin-786602 |



| | |
|---|---|
| Name : | Kaustubh P Gholap |
| Mobile Number : | 7798129513 |
| E-mail : | kaustubhpramod.gholap2019@vitstudent.ac.in |
| Permanent Address: | Dhasai, Thane, Maharashtra, India, Pin-421402 |

| | |
|---|---|
| Name : | Sam Methuselah |
| Mobile Number : | 9755191037 |
| E-mail : | sam.methuselah2019@vitstudent.ac.in |
| Permanent Address: | D-50/2, RRCAT Colony, Indore, M.P., Pin-452013 |

| | |
|---|---|
| Name : | Arjun Arora |
| Mobile Number : | 9650383570 |
| E-mail : | arjun.arora2019@vitstudent.ac.in |
| Permanent Address: | 188, Kohat Enclave, Pitampura, Delhi |