CEC Continuing Education Center

DesignNews

IoT Designs Using STmicro Microcontrollers

# Day 4:

# Nucleo-C071RB Networking

Sponsored by

DigiKey

informa markets

# Webinar Logistics

- Turn on your system sound to hear the streaming presentation.
- If you have technical problems, click “Help” or submit a question asking for assistance.
- Participate in ‘Attendee Chat’ by maximizing the chat widget in your dock.

# Fred Eady

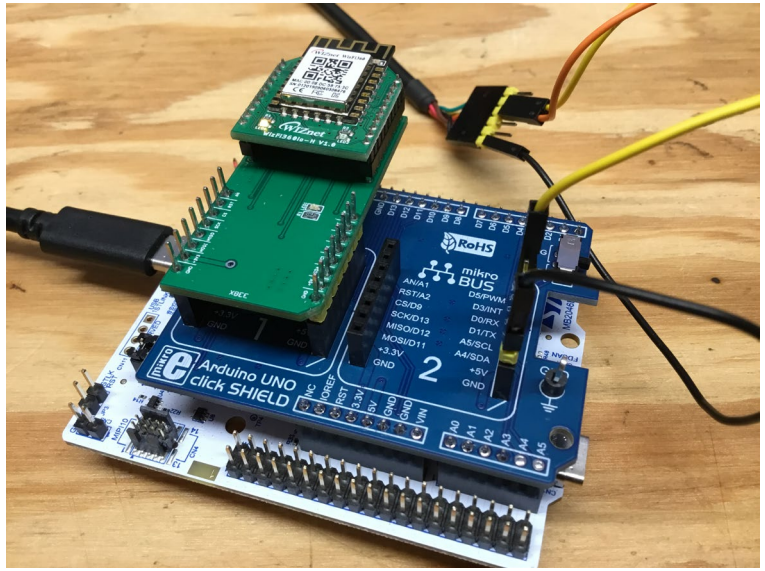
Visit ‘Lecturer Profile’ in your console for more details.

CEC Continuing Education Center

# AGENDA

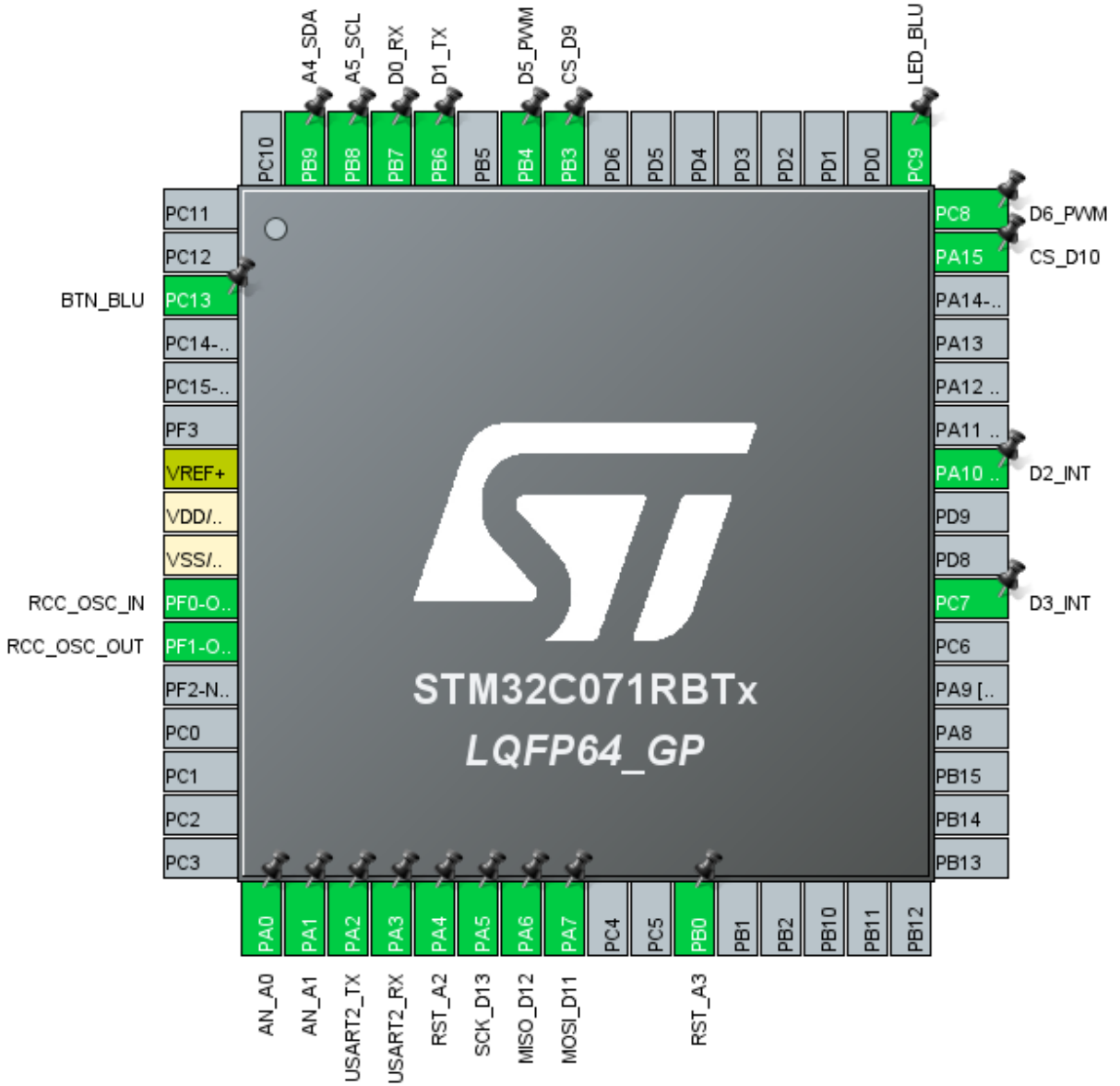
DigiKey

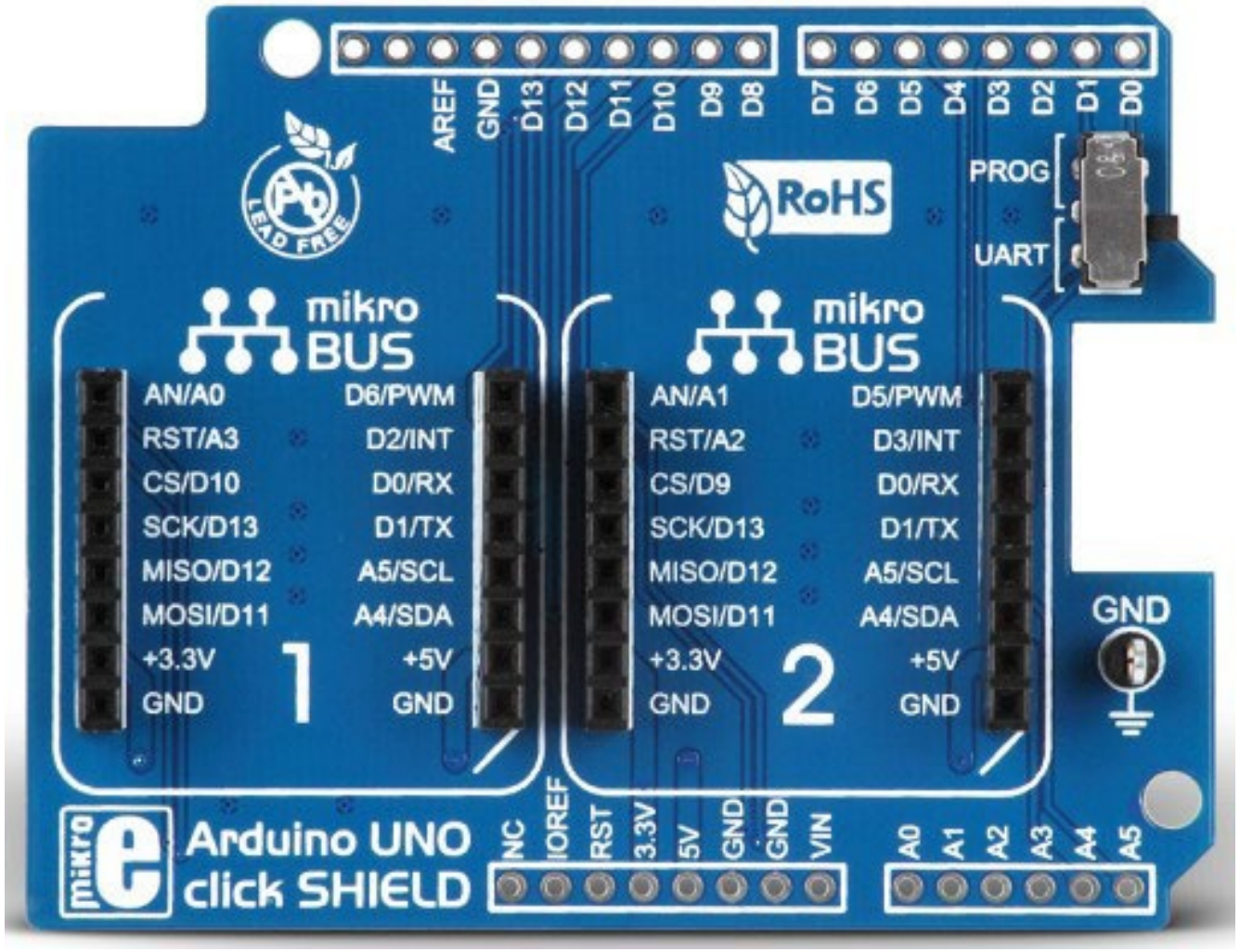
- Build a NUCLEO-C071RB ETH WIZ click Application
  - Configure the Hardware
  - Code the DHCP Module
  - Code the Connect Module
  - Program and Run the ETH WIZ click Project
- Build a NUCLEO-C071RB WizFi360io-H Application
  - The Song Remains the Same



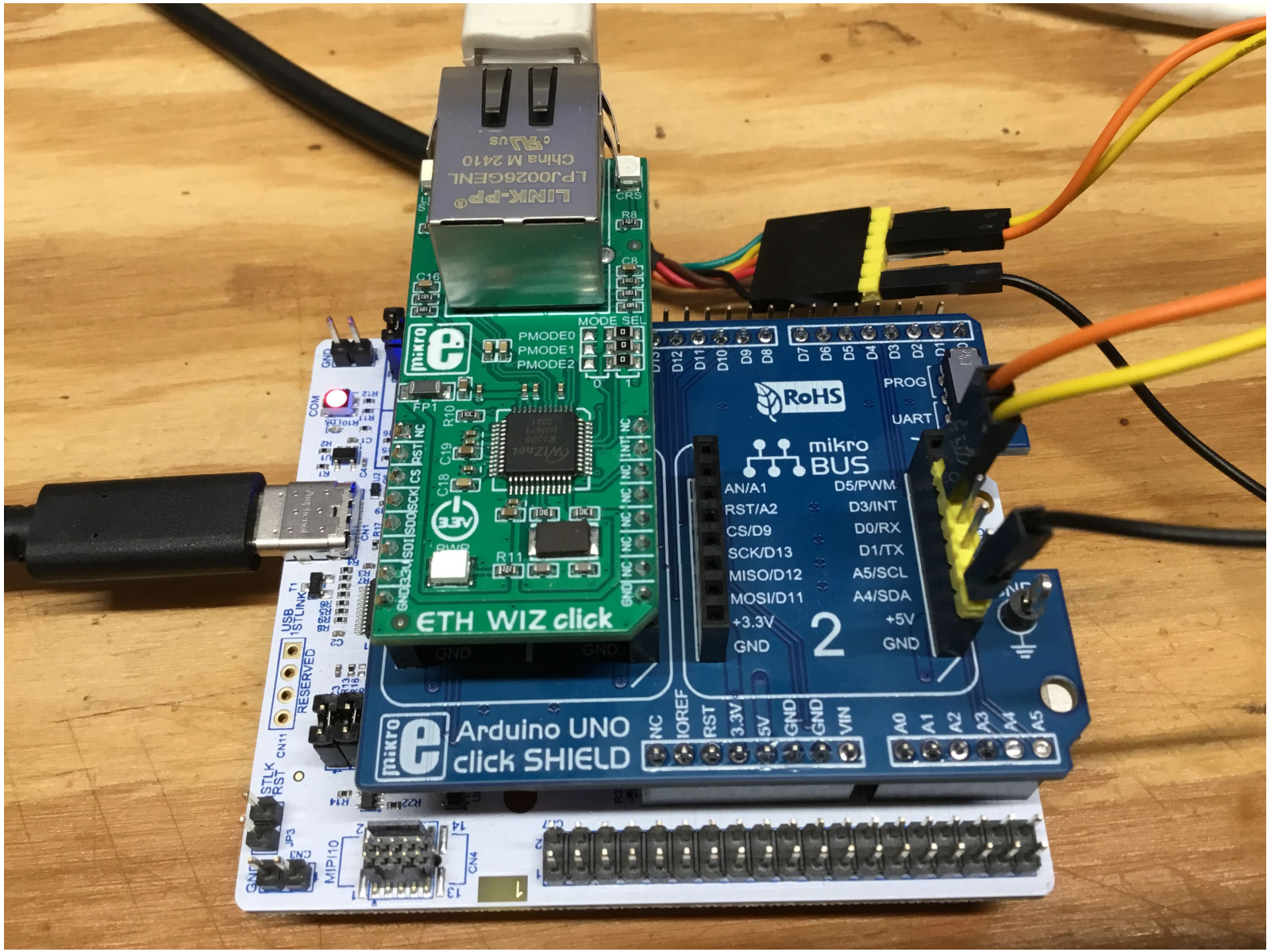


# Configure the Hardware



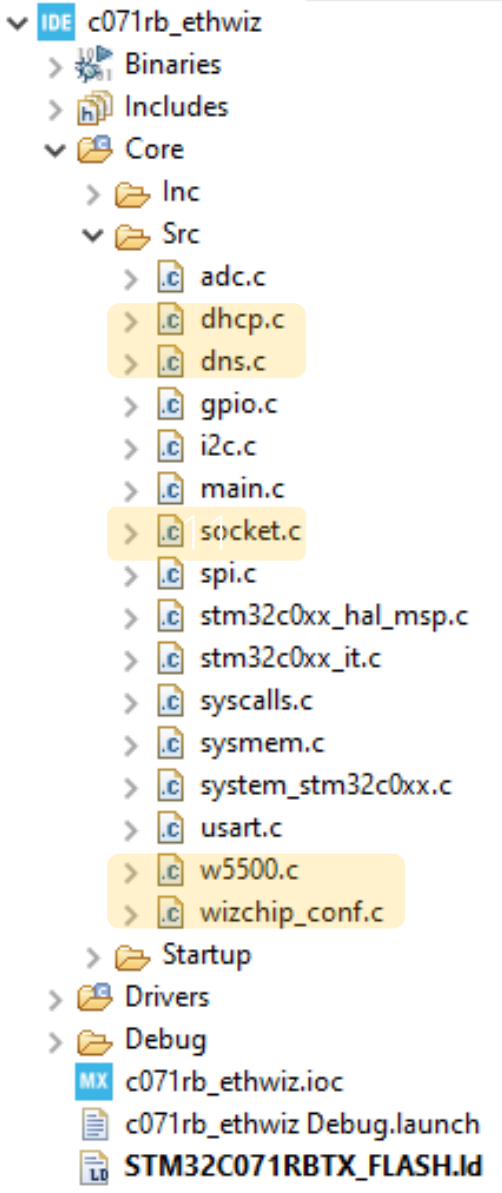



# Configure the Hardware



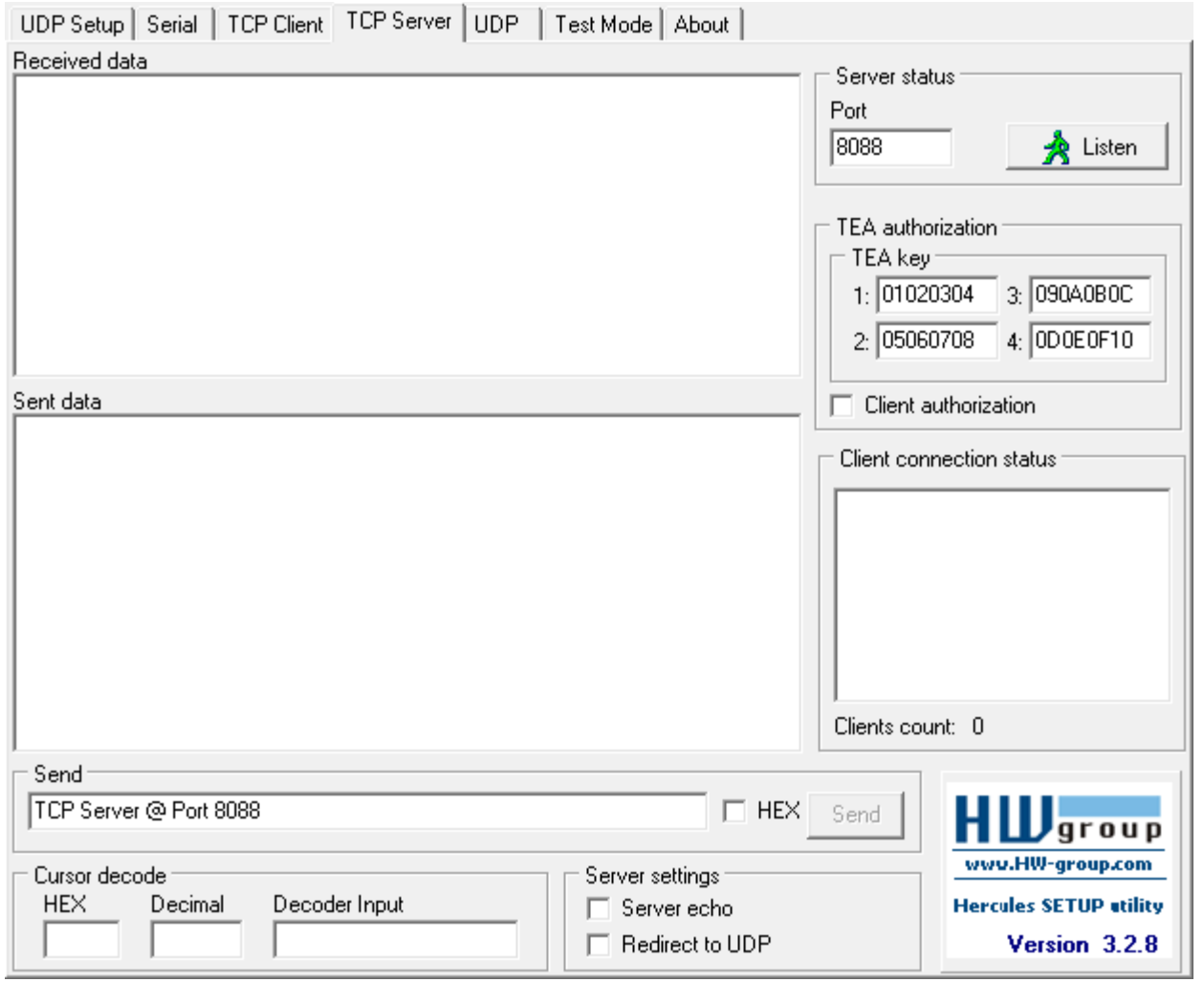
# Code the DHCP Module – Includes and Typedefs

```
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------
-*/
#include "main.h"
#include "adc.h"
#include "i2c.h"
#include "spi.h"
#include "usart.h"
#include "gpio.h"

/* Private includes ----------------------------------------------------------
-*/
/* USER CODE BEGIN Includes */
#include <stdio.h>
#include "wizchip_conf.h"
#include "socket.h"
#include "dhcp.h"
#include <string.h>
#include <stdint.h>
/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------
-*/
/* USER CODE BEGIN PTD */
wiz_NetInfo netInfo = { .mac = {0x00,0x04,0xA3,0x06,0xE7,0x48},
                        .dhcp = NETINFO_DHCP
};
// pstates
enum{
  MODE_DHCP = 0,
  MODE_CONNECT,
};
/* USER CODE END PTD */
```



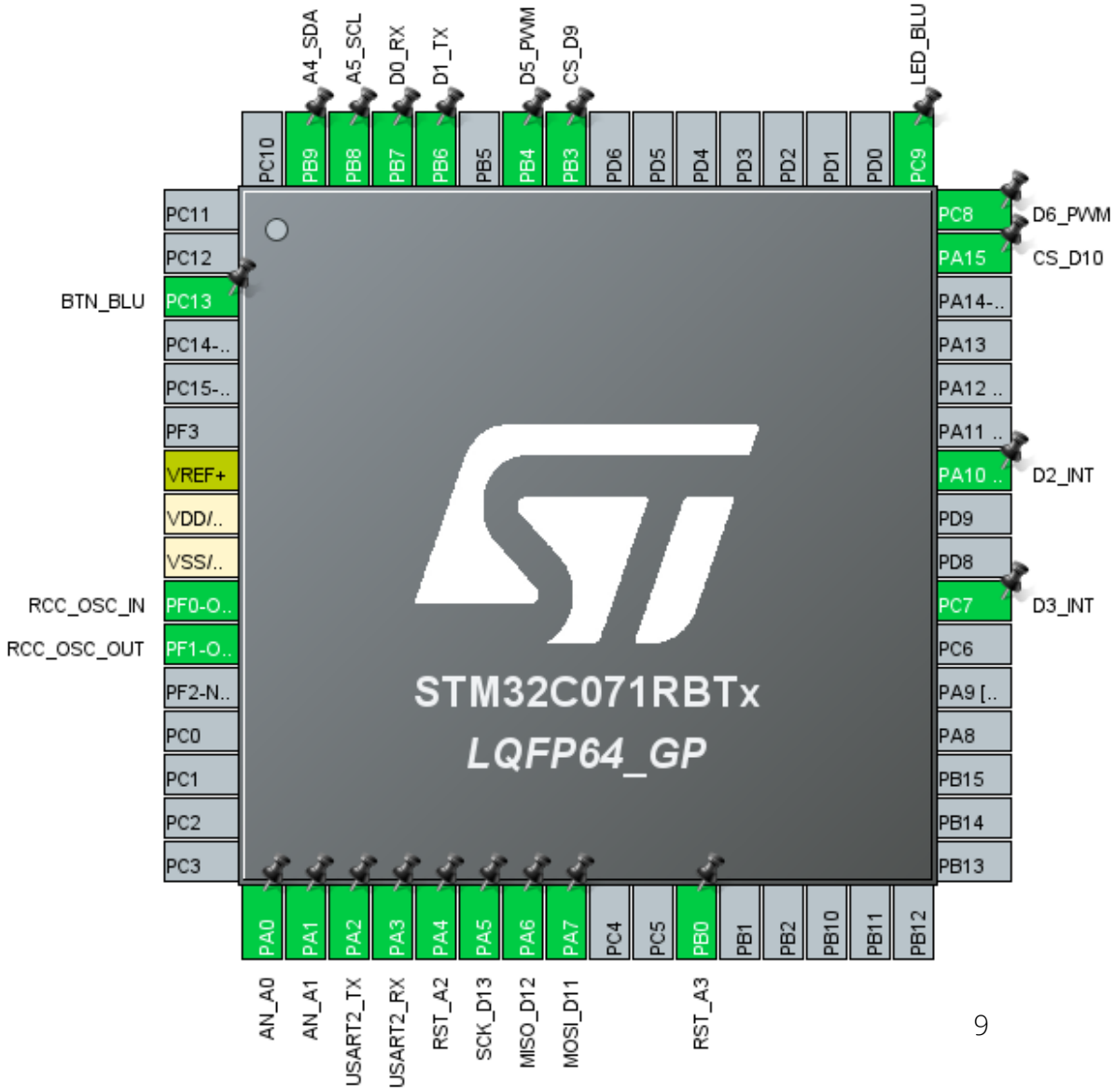
# Code the DHCP Module – Variables

```
/* USER CODE BEGIN PV */
//*****************************************************************
//* SOCKET USER VARIABLES
//*****************************************************************
uint8_t sktRxBuf[64];
uint8_t retVal;
int16_t rcvLen;
uint8_t bufSize[] = {2, 2, 2, 2};
uint8_t dhcpBuf[1024];
uint8_t serverIP[] = {0xC0,0xA8,0x01,0xF4}; //192.168.1.244
uint16_t serverPort = 8088;
//*****************************************************************
//* USER VARIABLES
//*****************************************************************
uint8_t scratch8;
uint8_t pstate;
uint8_t lastpstate;
/* USER CODE END PV */
```



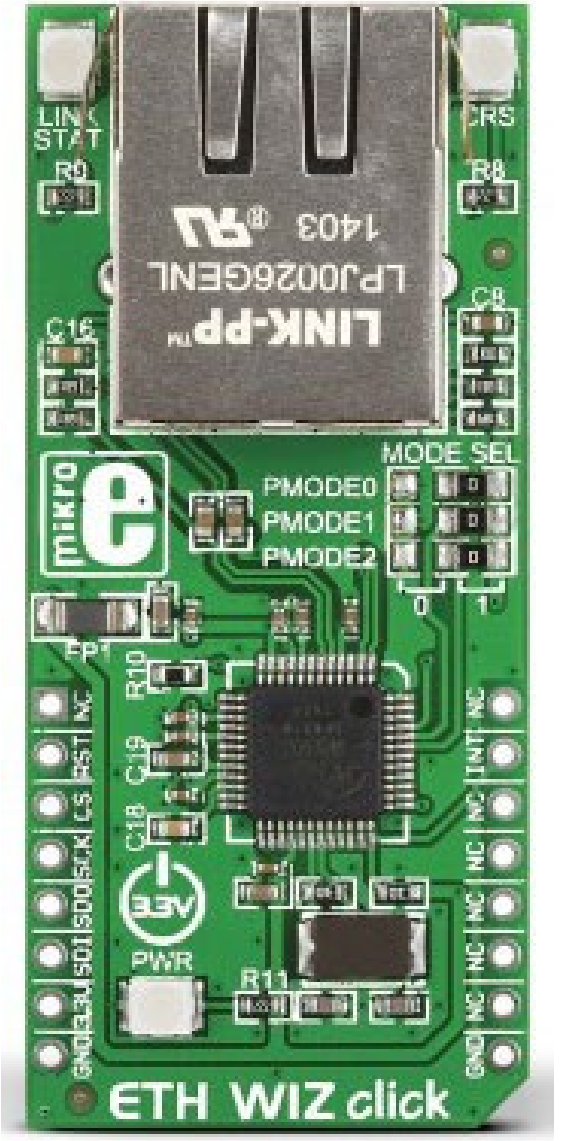
# Code the DHCP Module – SPI Functions

```
//*****************************************************************************
//* WIZNET SPI CS FUNCTIONS
//*****************************************************************************
void wiz_csLO(void)
{
  HAL_GPIO_WritePin(CS_D10_GPIO_Port, CS_D10_Pin, GPIO_PIN_RESET);
}
void wiz_csHI(void)
{
  HAL_GPIO_WritePin(CS_D10_GPIO_Port, CS_D10_Pin, GPIO_PIN_SET);
}
//*****************************************************************************
//* SPI READ/WRITE FUNCTIONS
//*****************************************************************************
void ReadBuf(uint8_t* buf, uint16_t len)
{
  HAL_SPI_Receive(&hspi1,buf,len,HAL_MAX_DELAY);
}
void WriteBuf(uint8_t* buf,uint16_t len)
{
  HAL_SPI_Transmit(&hspi1,buf,len,HAL_MAX_DELAY);
}
uint8_t ReadByte(void)
{
  uint8_t bite;
  ReadBuf(&bite,sizeof(bite));
  return bite;
}
void WriteByte(uint8_t bite)
{
  WriteBuf(&bite,sizeof(bite));
}
```



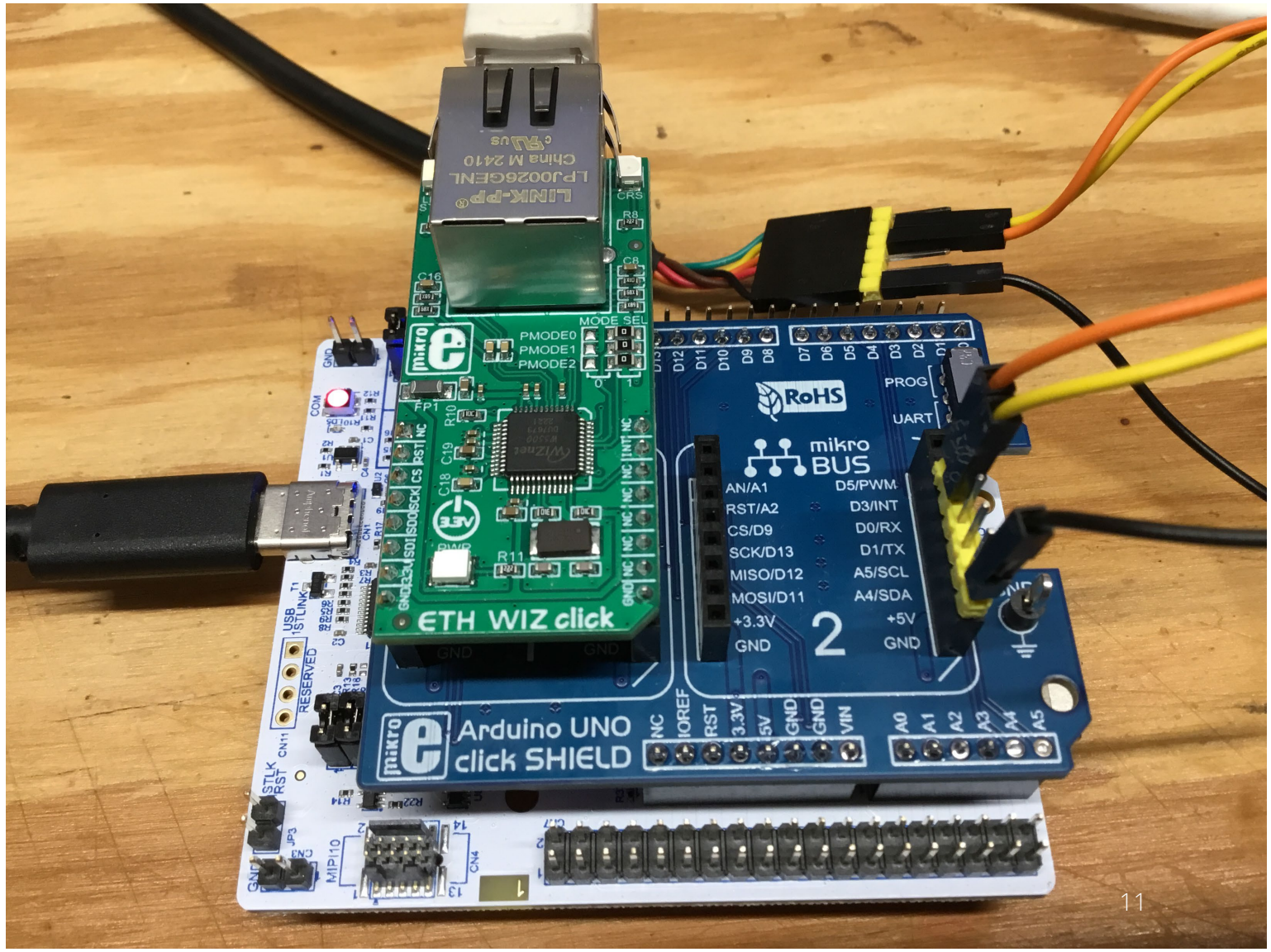
# Code the DHCP Module – Register Callbacks

```
printf("INFO: Register Callbacks\r\n");
reg_wizchip_cs_cbfunc(wiz_csLO, wiz_csHI);
reg_wizchip_spi_cbfunc(ReadByte, WriteByte);
reg_wizchip_spiburst_cbfunc(ReadBuf,WriteBuf);
printf("INFO: Calling wizchip_init\r\n");
wizchip_init(bufSize, bufSize); // Initialize Socket Buffer Size
setSHAR(netInfo.mac); // Set the local MAC Address


printf("INFO: DHCP_Init\r\n");
DHCP_init(0,dhcpBuf);
```



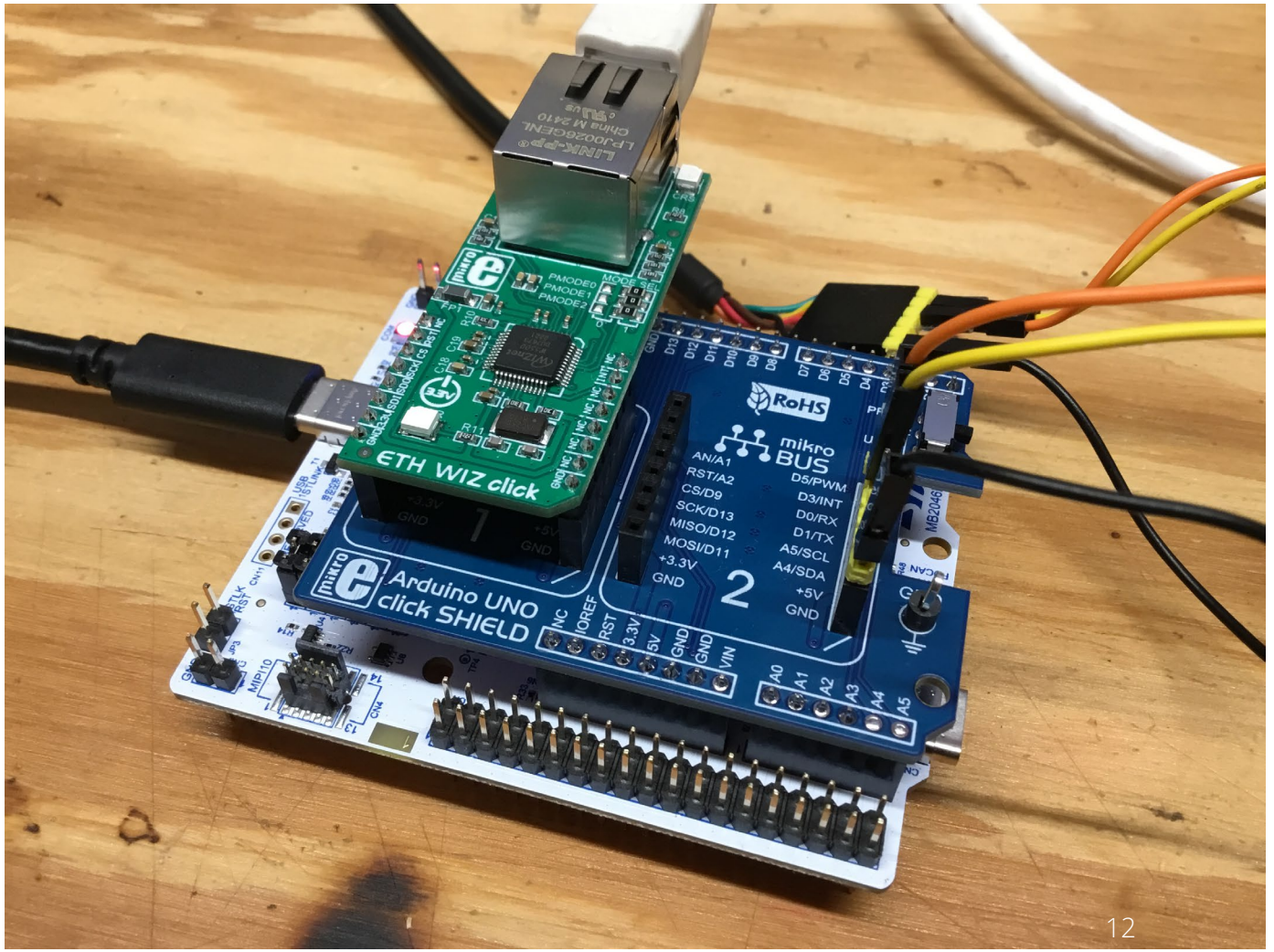
# Code the DHCP Module

```
//*************************************************
//* MODE_DHCP
//*************************************************
case MODE_DHCP:
 switch(DHCP_run())
 {
   case DHCP_FAILED:
    printf("DHCP FAILED\r\n");
   break;
   case DHCP_RUNNING:
   break;
   case DHCP_IP_ASSIGN:
    printf("INFO: DHCP IP ASSIGN\r\n");
   break;
   case DHCP_IP_CHANGED:
    printf("INFO: DHCP IP CHANGED\r\n");
   break;
   case DHCP_IP_LEASED:
    printf("INFO: DHCP IP LEASED\r\n");
    wizchip_getnetinfo(&netInfo);
    showNetInfo();
    wizchip_setnetinfo(&netInfo);
    pstate = MODE_CONNECT;
   break;
   case DHCP_STOPPED:
    printf("INFO: DHCP STOPPED\r\n");
   break;
 }
break;
```

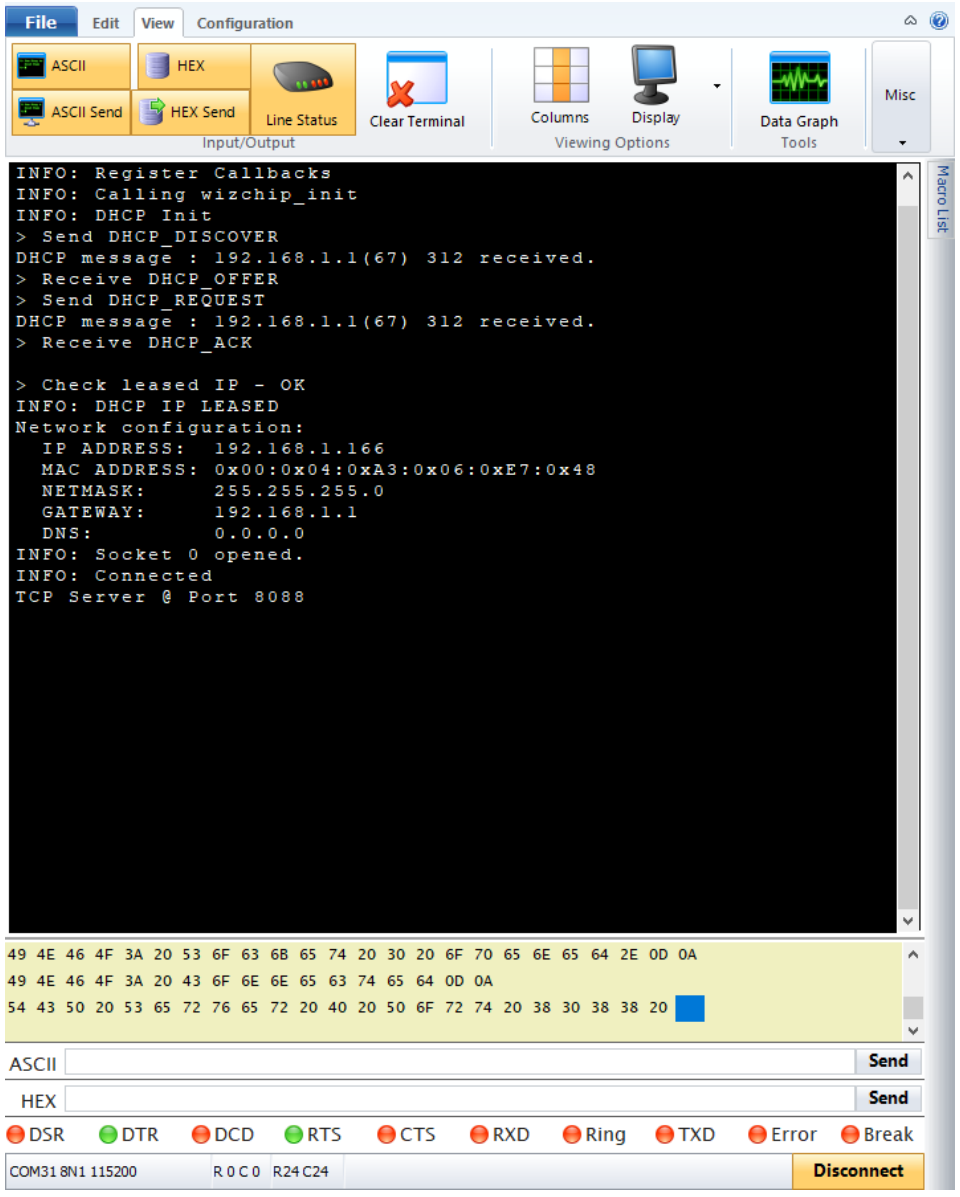


# Code the Connect Module

```
//************************************************************
//* MODE_CONNECT
//************************************************************
case MODE_CONNECT:
 retVal = socket(0, Sn_MR_TCP, 5000, 0);
 if(retVal == 0)
 {
  printf("INFO: Socket 0 opened.\r\n");
  retVal = connect(0, serverIP, 8088);
  if(retVal == SOCK_OK)
  {
   printf("INFO: Connected\r\n");
   HAL_Delay(2000);
   rcvLen = 0;
   do{
    rcvLen = getSn_RX_RSR(0);
   }while(rcvLen == 0);
   recv(0, sktRxBuf, rcvLen);
   HAL_UART_Transmit(&huart1, sktRxBuf, rcvLen, HAL_MAX_DELAY);
   while(1);
  }
 }
break;
```

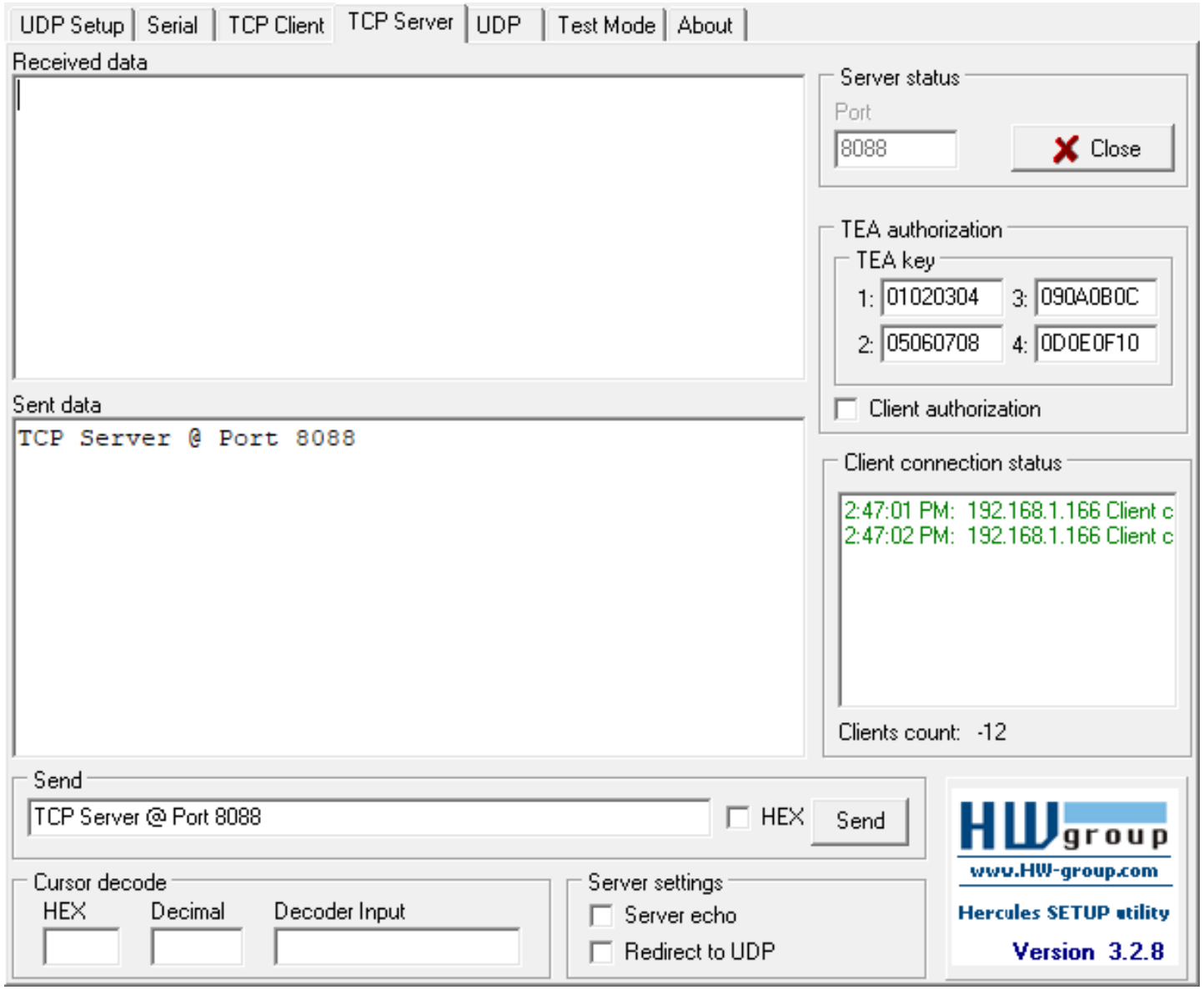


# Program and Run the ETH WIZ click Project



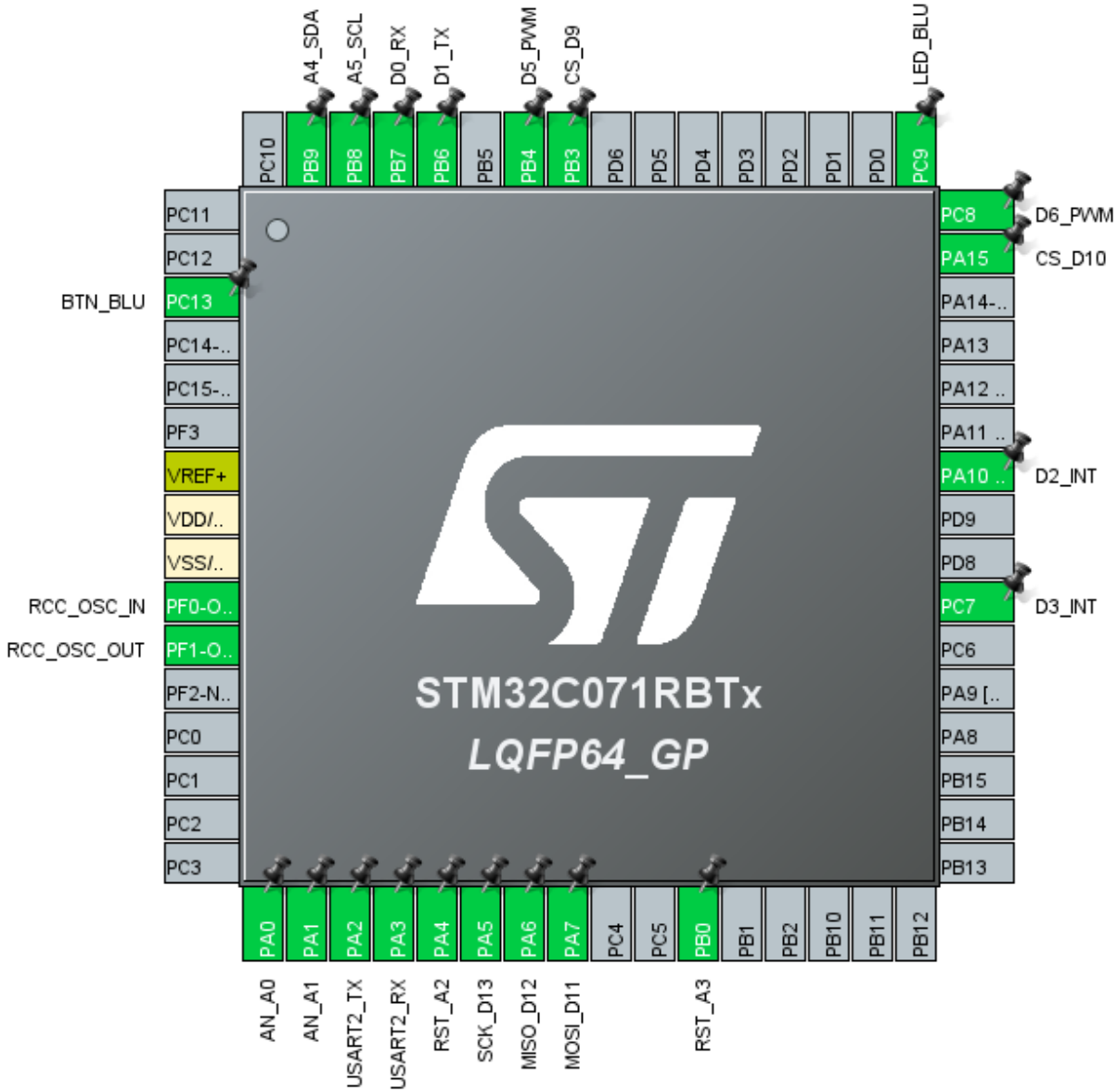

```
INFO: Register Callbacks
INFO: Calling wizchip_init
INFO: DHCP_Init
> Send DHCP_DISCOVER
DHCP message : 192.168.1.1(67) 312 received.
> Receive DHCP_OFFER
> Send DHCP_REQUEST
DHCP message : 192.168.1.1(67) 312 received.
> Receive DHCP_ACK

> Check leased IP - OK
INFO: DHCP IP LEASED
Network configuration:
   IP ADDRESS:   192.168.1.166
   MAC ADDRESS:  0x00:0x04:0xA3:0x06:0xE7:0x48
   NETMASK:      255.255.255.0
   GATEWAY:      192.168.1.1
   DNS:          0.0.0.0
INFO: Socket 0 opened.
INFO: Connected
TCP Server @ Port 8088
```

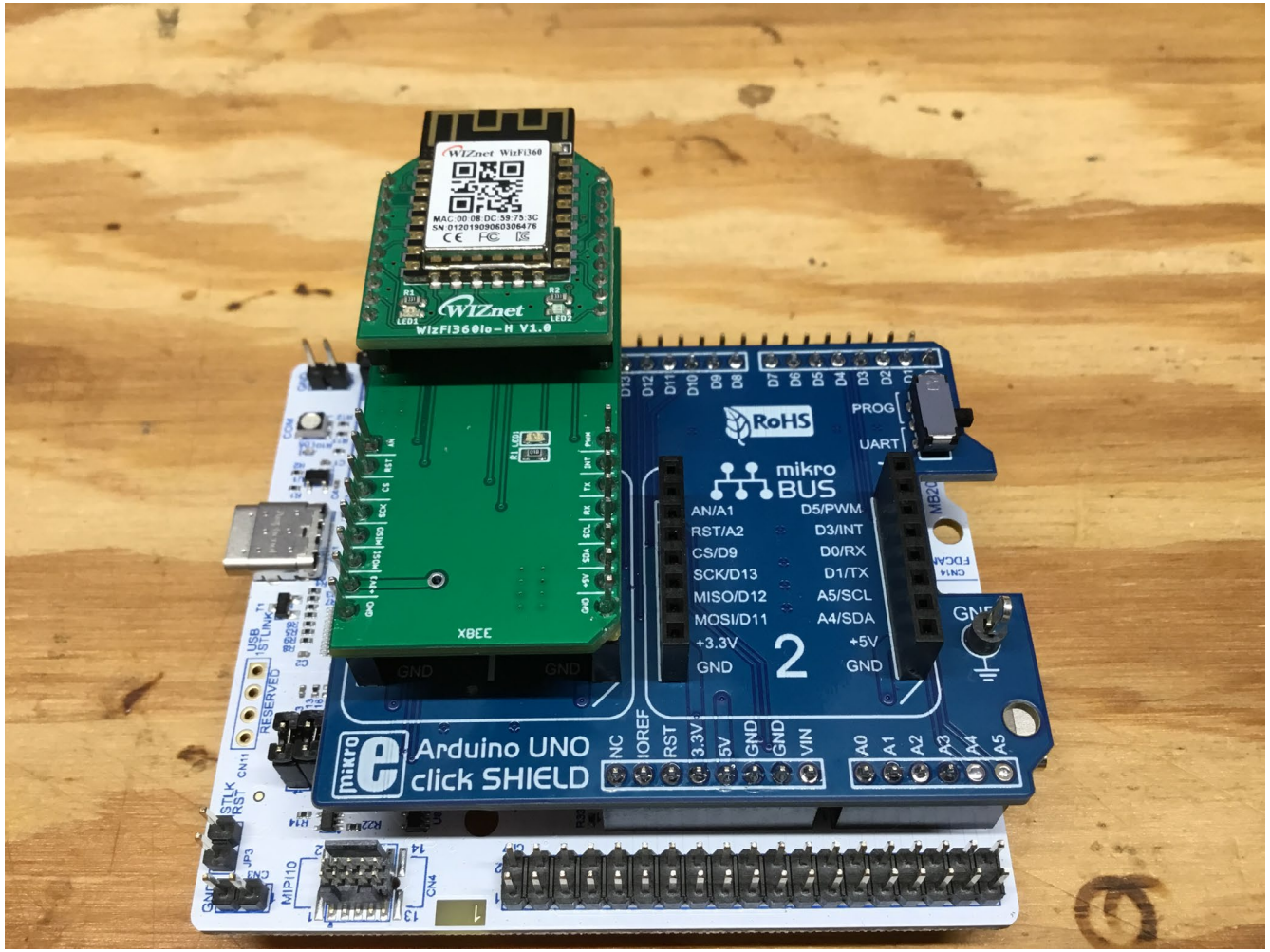
# Program and Run the ETH WIZ click Project




| Expression | Type | Value |
|---|---|---|
| sktRxBuf | uint8_t [64] | [64] |
| (x)= sktRxBuf[0] | uint8_t | 84 'T' |
| (x)= sktRxBuf[1] | uint8_t | 67 'C' |
| (x)= sktRxBuf[2] | uint8_t | 80 'P' |
| (x)= sktRxBuf[3] | uint8_t | 32 ' ' |
| (x)= sktRxBuf[4] | uint8_t | 83 'S' |
| (x)= sktRxBuf[5] | uint8_t | 101 'e' |
| (x)= sktRxBuf[6] | uint8_t | 114 'r' |
| (x)= sktRxBuf[7] | uint8_t | 118 'v' |
| (x)= sktRxBuf[8] | uint8_t | 101 'e' |
| (x)= sktRxBuf[9] | uint8_t | 114 'r' |
| (x)= sktRxBuf[10] | uint8_t | 32 ' ' |
| (x)= sktRxBuf[11] | uint8_t | 64 '@' |
| (x)= sktRxBuf[12] | uint8_t | 32 ' ' |
| (x)= sktRxBuf[13] | uint8_t | 80 'P' |
| (x)= sktRxBuf[14] | uint8_t | 111 'o' |
| (x)= sktRxBuf[15] | uint8_t | 114 'r' |
| (x)= sktRxBuf[16] | uint8_t | 116 't' |
| (x)= sktRxBuf[17] | uint8_t | 32 ' ' |
| (x)= sktRxBuf[18] | uint8_t | 56 '8' |
| (x)= sktRxBuf[19] | uint8_t | 48 '0' |
| (x)= sktRxBuf[20] | uint8_t | 56 '8' |
| (x)= sktRxBuf[21] | uint8_t | 56 '8' |
| (x)= sktRxBuf[22] | uint8_t | 32 ' ' |

# Configure the Hardware



# Code the Application – AT Handlers

```
/* USER CODE BEGIN PFP */
void sendAT(void);
uint8_t chk_atok(void);
uint8_t setStationMode(void);
uint8_t setSingleConnectionMode(void);
uint8_t setDhcpEnable(void);
uint8_t connect2AP(void);
uint8_t connect2Server(void);
uint8_t send2Server(void);
/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
//*****************************************************************************
//* SEND AT
//*****************************************************************************
void sendAT(void)
{
  uint8_t cmd[]={'A','T','\r','\n'};
  HAL_UART_Transmit(&huart1, cmd, sizeof(cmd), HAL_MAX_DELAY);
}
```

# Code the Application – AT Handlers

```
//*****************************************************************************
//* CHECK FOR OK
//*****************************************************************************
uint8_t chk_atok(void)
{
 rc = 0;
 bufIndx = 0;
 memset(rxBuf,0,sizeof(rxBuf));
 HAL_GPIO_WritePin(LED_BLU_GPIO_Port,LED_BLU_Pin,GPIO_PIN_SET);
 do{
    if(IsCharInQueue())
    {
     rxBuf[bufIndx++] = readRxRing();
    }
   }while(IsCharInQueue());
 if(rxBuf[bufIndx-4] == 'O' && rxBuf[bufIndx-3] == 'K')
 {
  rc = 1;
 }
 return rc;
}
```

# Code the Application – Typical Command Sequence

```
//*****************************************************************************
//* SET SINGLE CONNECTION MODE
//*****************************************************************************
uint8_t setSingleConnectionMode(void)
{
 uint8_t cmd[]={'A','T','+','C','I','P','M','U','X','=','0','\r','\n'};
 rc = 0;
 resetRxRing();
 //printf("AT+CIPMUX=0\r\n");
 HAL_UART_Transmit(&huart1, cmd, sizeof(cmd), HAL_MAX_DELAY);
 HAL_Delay(1000);
 if(chk_atok())
 {
  rc = 1;
 }
 return rc;
}
```

# Code the Application - Initialization

```
RingBuffer_Init();
HAL_GPIO_WritePin(RST_A3_GPIO_Port,RST_A3_Pin,GPIO_PIN_RESET);
HAL_Delay(500);
HAL_GPIO_WritePin(RST_A3_GPIO_Port,RST_A3_Pin,GPIO_PIN_SET);
HAL_Delay(500);
resetRxRing();
do
{
 resetRxRing();
 bufIndx = 0x00;
 //printf("AT\r\n");
 sendAT();
 HAL_Delay(50);
 do{
  if(IsCharInQueue())
  {
   rxBuf[bufIndx++] = readRxRing();
  }
 }while(IsCharInQueue());
 if(rxBuf[bufIndx-4] == 'O' && rxBuf[bufIndx-3] == 'K’)
 {
  okat = 0x01;
 }
}while(okat == 0);
```

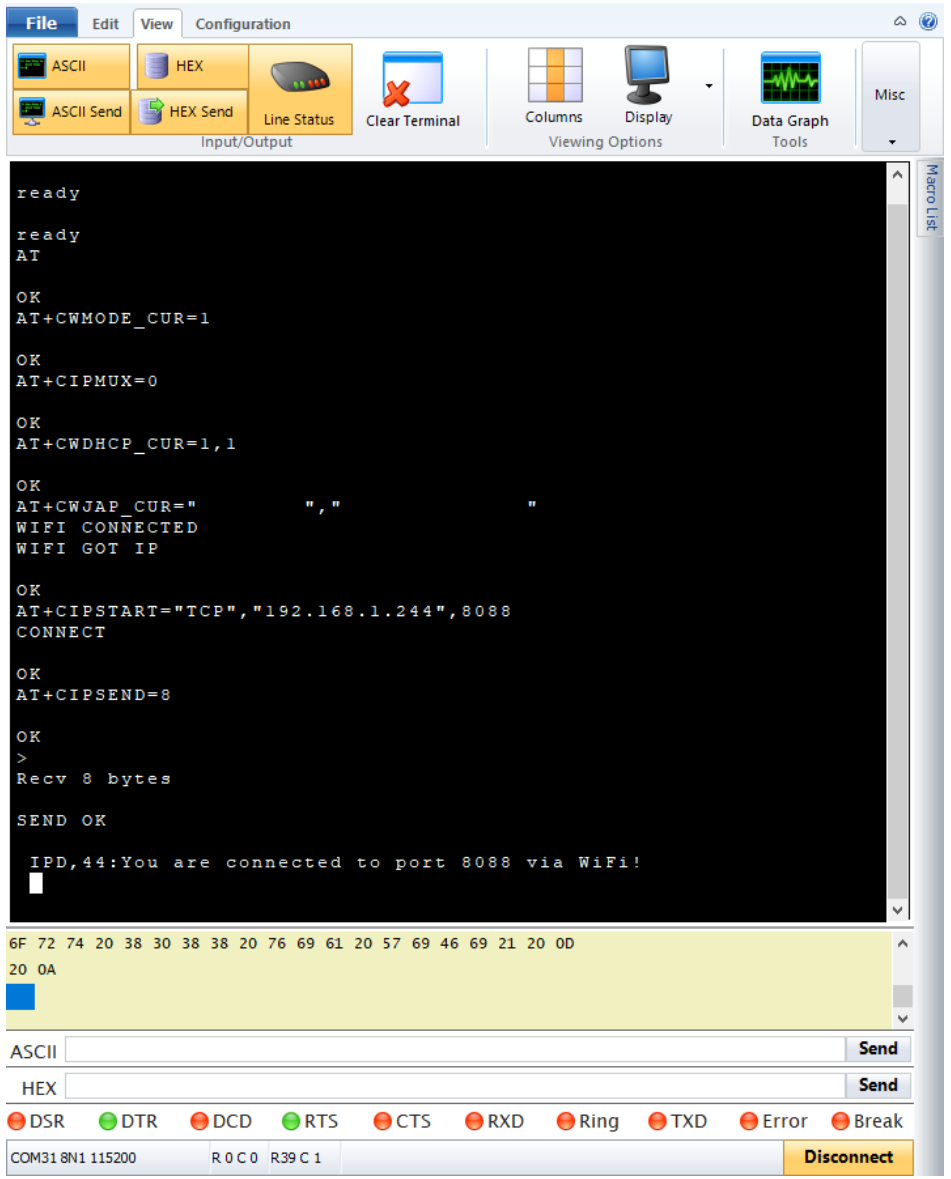
# Code the Application – Application Flow

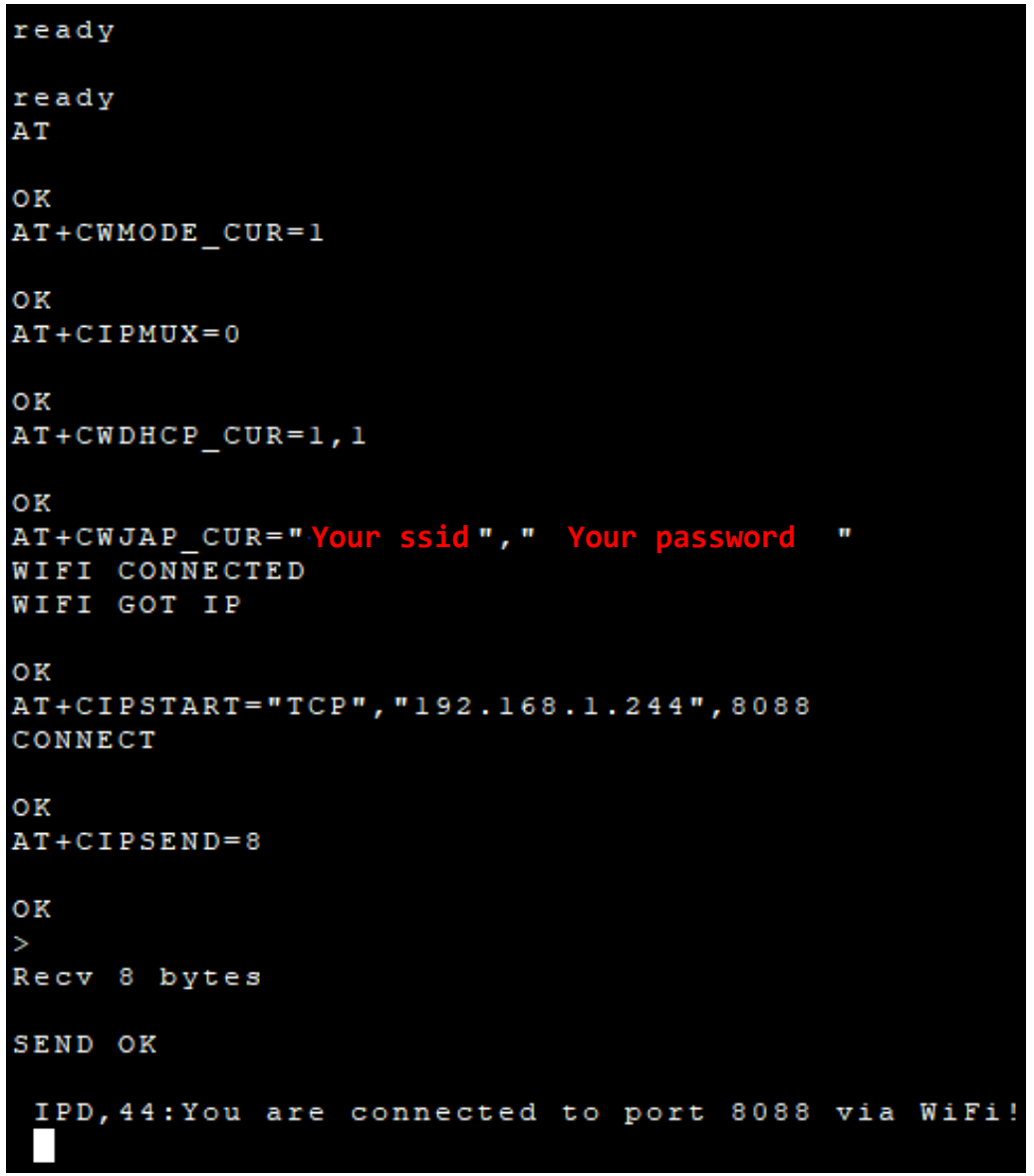
```
if(!setStationMode())
{
 do{
  HAL_GPIO_WritePin(LED_BLU_GPIO_Port,LED_BLU_Pin,GPIO_PIN_SET);
  Delay(200);
  HAL_GPIO_WritePin(LED_BLU_GPIO_Port,LED_BLU_Pin,GPIO_PIN_RESET);
  HAL_Delay(200);
  }while(1);
}

if(!setSingleConnectionMode())
{
do{
  HAL_GPIO_WritePin(LED_BLU_GPIO_Port,LED_BLU_Pin,GPIO_PIN_SET);
  HAL_Delay(100);
  HAL_GPIO_WritePin(LED_BLU_GPIO_Port,LED_BLU_Pin,GPIO_PIN_RESET);
  HAL_Delay(100);
  }while(1);
}

if(!setDhcpEnable())
{
  do{
  HAL_GPIO_WritePin(LED_BLU_GPIO_Port,LED_BLU_Pin,GPIO_PIN_SET);
  HAL_Delay(100);
  HAL_GPIO_WritePin(LED_BLU_GPIO_Port,LED_BLU_Pin,GPIO_PIN_RESET);
  HAL_Delay(100);
  }while(1);
}
```

# Code the Application – Send Some Data to the Server

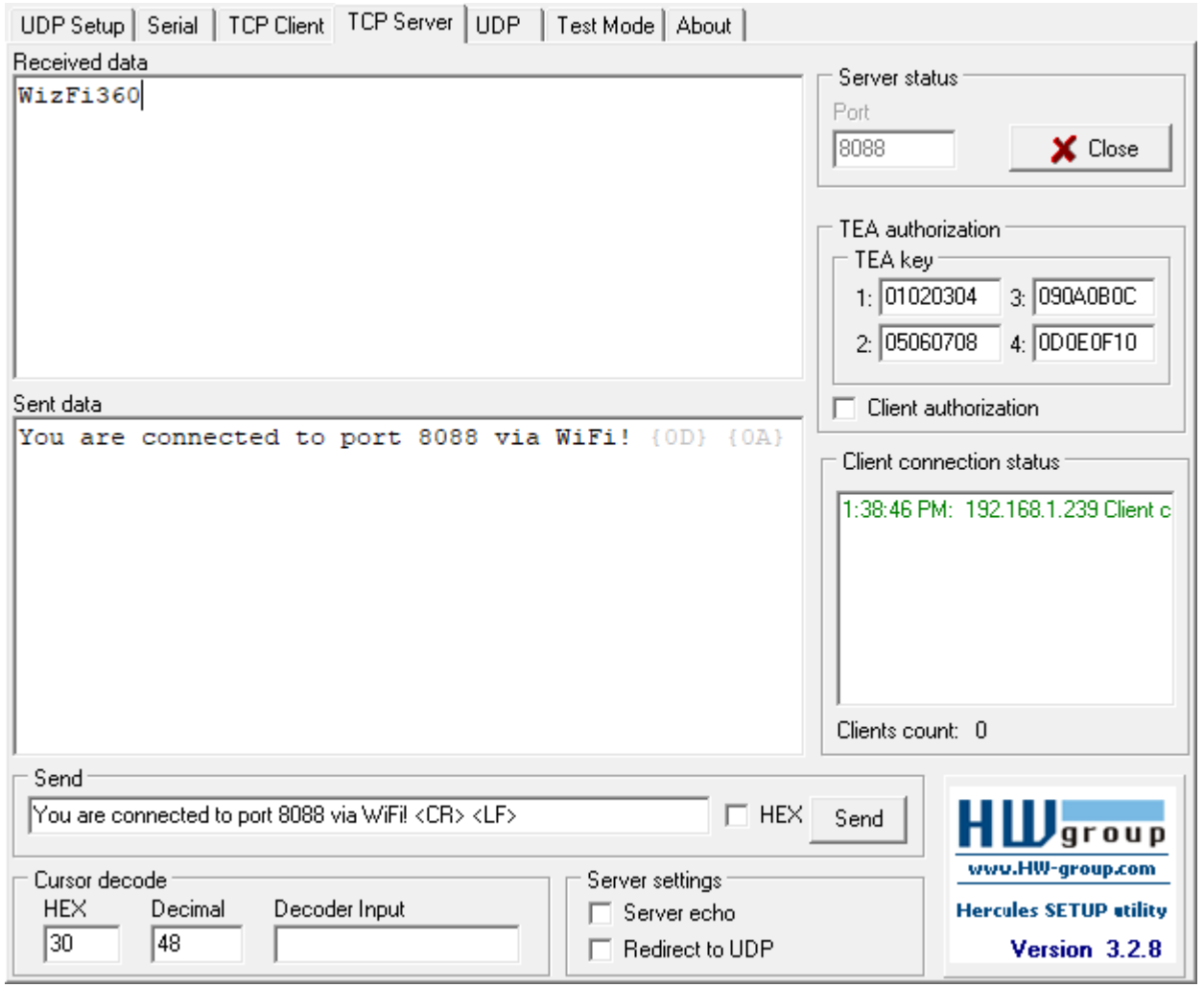
```
//*****************************************************************************
//* SEND DATA TO THE SERVER
//*****************************************************************************
uint8_t send2Server(void)
{
 uint8_t bite[] = {'W','i','z','F','i','3','6','0’};
 uint8_t cmd[]={'A','T','+','C','I','P','S','E','N','D','=','8','\r','\n’};
 rc = 0;
 resetRxRing();
 memset(rxBuf,0x00,sizeof(rxBuf));
 bufIndx = 0;
 //printf("AT+CIPSEND=8\r\n");
 HAL_UART_Transmit(&huart1, cmd, sizeof(cmd), HAL_MAX_DELAY);
 HAL_Delay(1000);

 do{
  if(IsCharInQueue())
  {
   rxBuf[bufIndx++] = readRxRing();
  }
 }while(IsCharInQueue());

 if(rxBuf[bufIndx-2] == '>’)
 {
  resetRxRing();
  memset(rxBuf,0x00,sizeof(rxBuf));
  bufIndx = 0;
  HAL_UART_Transmit(&huart1, bite, 8, HAL_MAX_DELAY);
  HAL_Delay(2000);
```

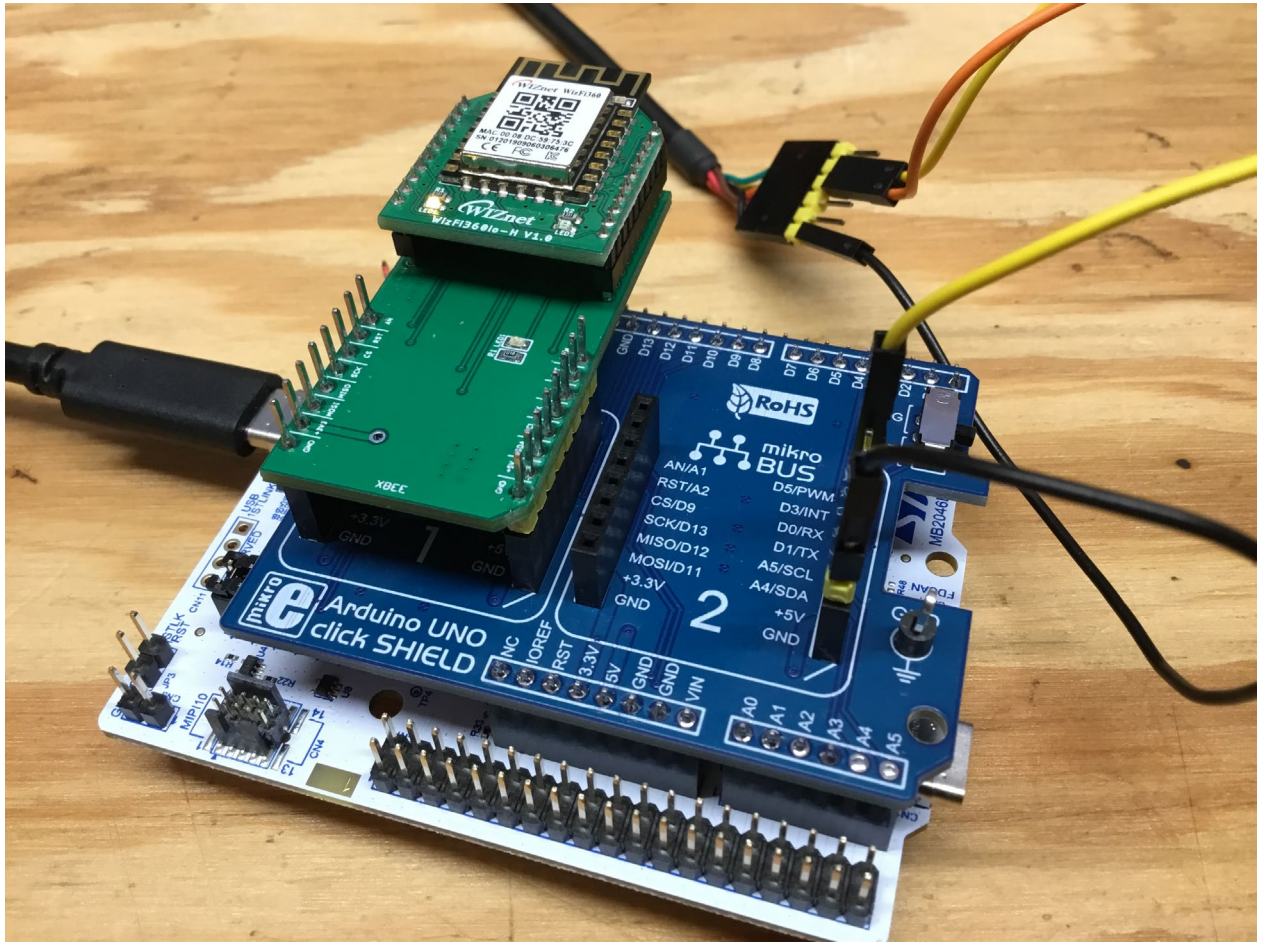
# Program and Run the WizFi360 Application



# Program and Run the WizFi360 Application



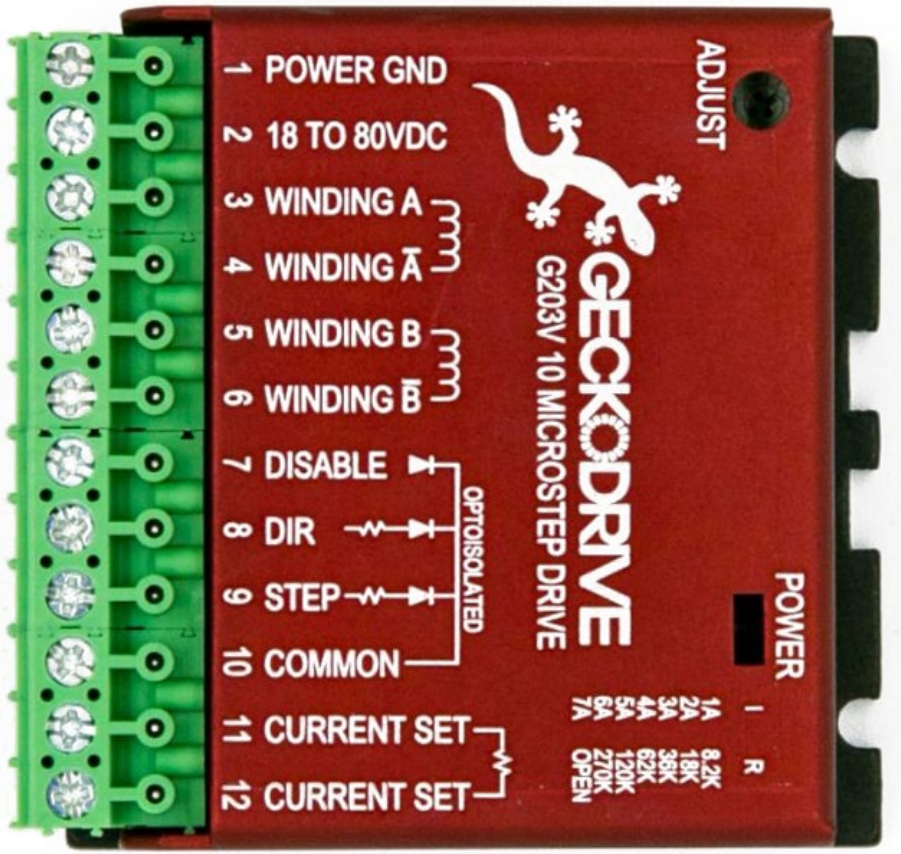
## Next Time…

MORE TO COME..

# Thank you for attending!!!

## Please consider the resources below:

- Today's Download Package
- STM32C071RB Datasheet
- NUCLEO-C071RB Schematic
- WizFi360io-H User Manual



CEC Continuing Education Center

DesignNews

# Thank You

Sponsored by

DigiKey

informa markets