Essential Coding Techniques for Hardware Engineers

The Data's in the Mail

February 1, 2019

Fred Eady

DesignNews

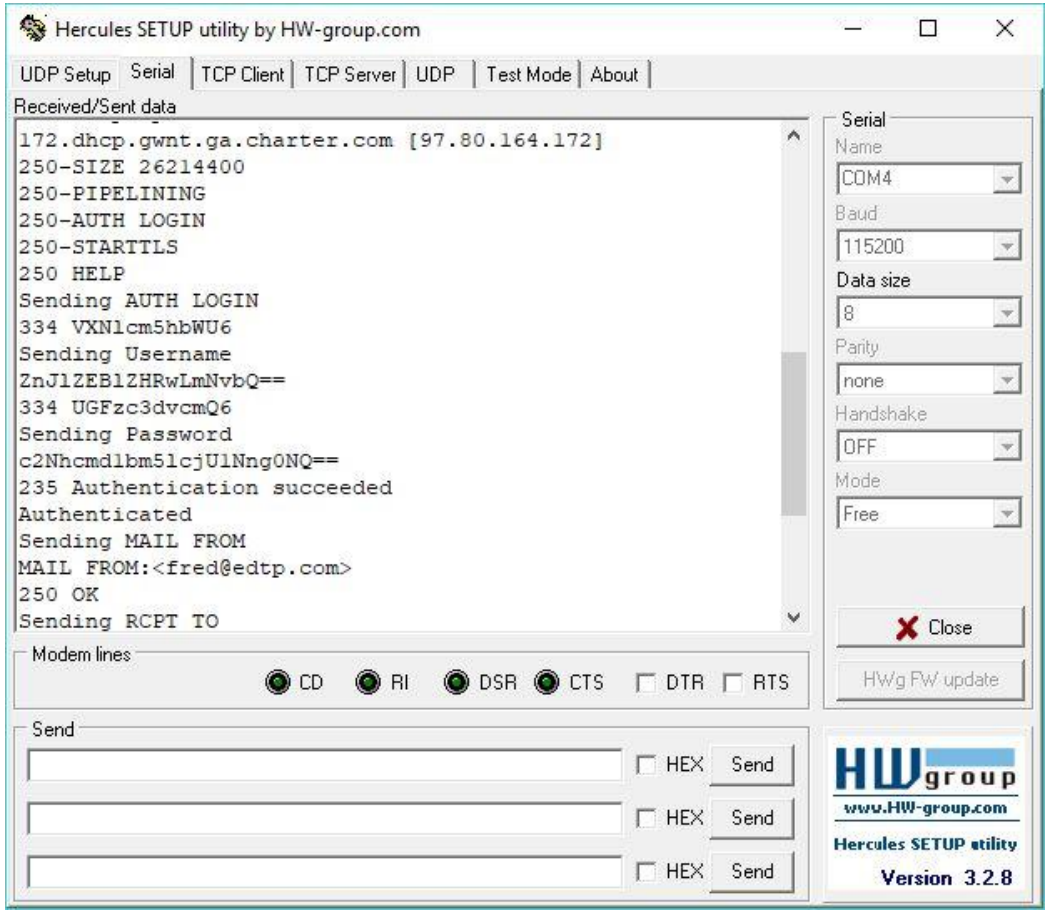
Presented by:

CEC CONTINUING EDUCATION CENTER

Digi-Key ELECTRONICS

AGENDA

- ARM SMTP Driver
- Day 5 Summary



Essential Coding Techniques for Hardware Engineers

ARM SMTP Driver: Enumerations and Function Prototypes

```
61  //*****************************************************************************
62  //* FUNCTION PROTOTYPES
63  //*****************************************************************************
64  void SystemClock_Config(void);
65  static void MX_GPIO_Init(void);
66  static void MX_SPI1_Init(void);
67  static void MX_USART1_UART_Init(void);
68  static void MX_USART2_UART_Init(void);
69  void getLCDpacket(void);
70  uint8_t CharInRing(void);
71  uint8_t readring(void);
72  void getMAC(void);
73  void mac_csLO(void);
74  void mac_csHI(void);
75  void eedata_csLO(void);
76  void eedata_csHI(void);
77  void wiz_csLO(void);
78  void wiz_csHI(void);
79  uint8_t spi_rb(void);
80  void spi_wb(uint8_t b);
81  void pageWR(uint16_t addr, uint8_t *buf, uint8_t *len);
82  void pageRD(uint16_t addr, uint8_t *buf, uint8_t *len);
83  void showNetInfo(void);
84  void base64_encode( unsigned char *dst, uint32_t dstlen, unsigned char *src, uint32_t srclen );
85  void base64_decode( unsigned char *dst, uint32_t dstlen, unsigned char *src, uint32_t srclen );
```

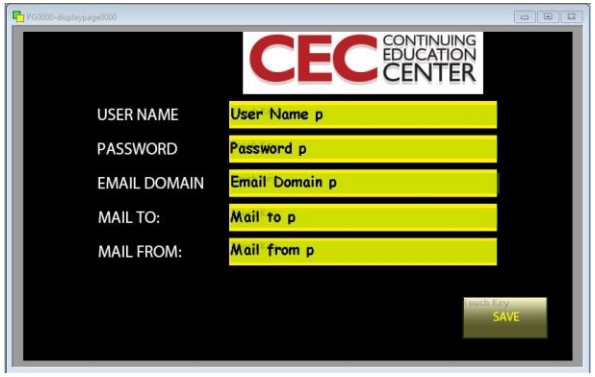
```
124 enum{
125     MODE_DISCOVERLCD = 0,
126     MODE_DATAENTRY,
127     MODE_NOLCD,
128     MODE_DHCP,
129     MODE_MONITOR,
130     MODE_CONNECT,
131     MODE_EHLO,
132     MODE_LOGIN,
133     MODE_AUTHENTICATE,
134     MODE_MAILFROM,
135     MODE_RCPTTO,
136     MODE_SENDDATA,
137     MODE_SENDMAIL,
138     MODE_QUIT
139 };
```


Essential Coding Techniques for Hardware Engineers

ARM SMTP Driver: Discover LCD Mode

```
case MODE_DISCOVERLCD:
  printf("MODE_DISCOVERLCD\r\n");
  for(scratch8=0;scratch8 < 15;scratch8++)
  {
    HAL_Delay(1000);
  }
  biteOut[0]  = 0xAA;
  biteOut[1]  = 0x30;
  biteOut[2]  = 0xCC;
  biteOut[3]  = 0x33;
  biteOut[4]  = 0xC3;
  biteOut[5]  = 0x3C;
  HAL_UART_Transmit(&huart1, biteOut, 6, 0xFFFF);

  for(scratch8=0;scratch8 < 10;scratch8++)
  {
    HAL_Delay(1000);
  }

  if(CharInRing())
  {
      rxBufLCD[0] = readring();
      if(rxBufLCD[0] == 0xAA)
      {
        rxBufLCD[1] = readring();
        if(rxBufLCD[1] == 0x30)
        {
          bufIndxLCD = 2;
          do{
              rxBufLCD[bufIndxLCD++] = readring();
              if(rxBufLCD[bufIndxLCD-1] == 0xCC)
              {
                rxBufLCD[bufIndxLCD++] = readring();
                if(rxBufLCD[bufIndxLCD-1] == 0x33)
                {
                  rxBufLCD[bufIndxLCD++] = readring();
                  if(rxBufLCD[bufIndxLCD-1] == 0xC3)
                  {
                    rxBufLCD[bufIndxLCD++] = readring();
                    if(rxBufLCD[bufIndxLCD-1] == 0x3C)
                    {
                      rxBufLCD[bufIndxLCD++] = readring();
                      if(rxBufLCD[bufIndxLCD-1] == 0x3A)
                      {
                        rxBufLCD[bufIndxLCD++] = readring();
                        if(rxBufLCD[bufIndxLCD-1] == 0x3E)
                        {
                          break;
                        }
```

```
    bufIndxLCD = 2;
    while(rxBufLCD[bufIndxLCD] != 'R')
      ++bufIndxLCD;

    bufIndx = 0;
    while(srcReady[bufIndx] == rxBufLCD[bufIndxLCD] && srcReady[bufIndx] != 0x00)
    {
      ++bufIndx;
      ++bufIndxLCD;
    }
    if(srcReady[bufIndx] == 0x00)
    {
      flags.fentry = 0;
      lastpstate = pstate;
      pstate = MODE_DATAENTRY;
    }
    else
    {
      lastpstate = pstate;
      pstate = MODE_NOLCD;
    }

  }
  else
  {
    lastpstate = pstate;
    pstate = MODE_NOLCD;
  }
break;
```



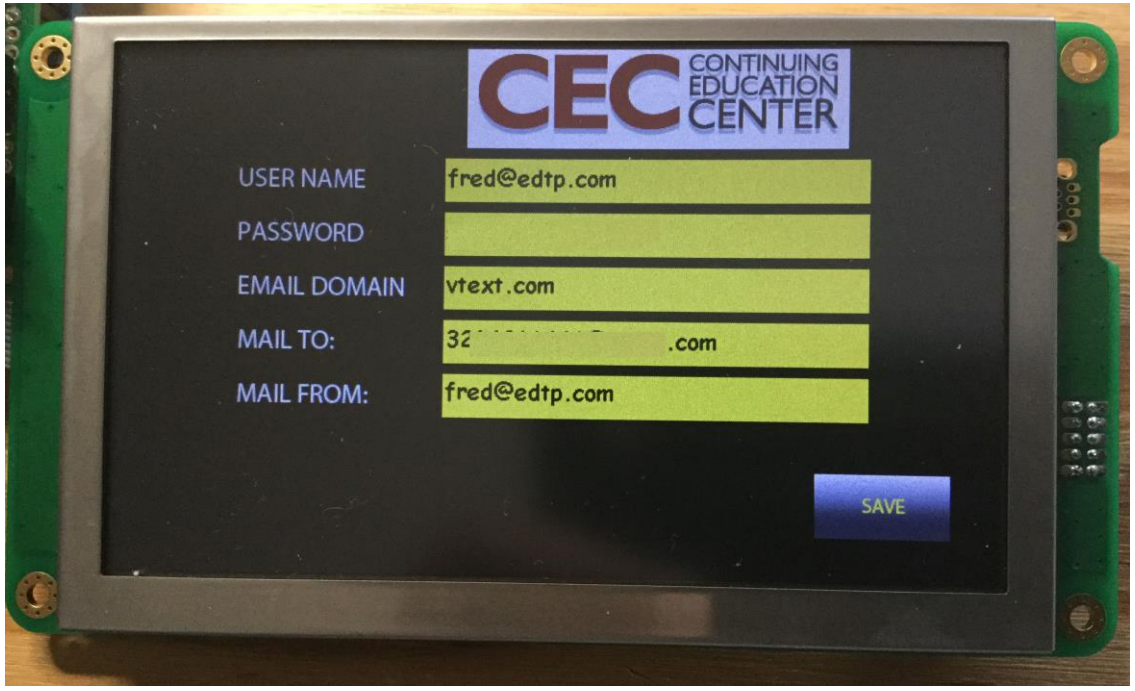

```
140 typedef struct{
141     uint16_t  fentry:1;
142     uint16_t  fmonitor:1;
143 }FFLAGS;
144 FFLAGS flags;
```

Essential Coding Techniques for Hardware Engineers

ARM SMTP Driver: Data Entry Mode

```
872     case MODE_DATAENTRY:
873       if(flags.fentry == 0)
874       {
875         printf("MODE_DATAENTRY\r\n");
876         flags.fentry = 1;
877       }
878         if(CharInRing())
879       {
880         getLCDpacket();
881       }
882     break;
```

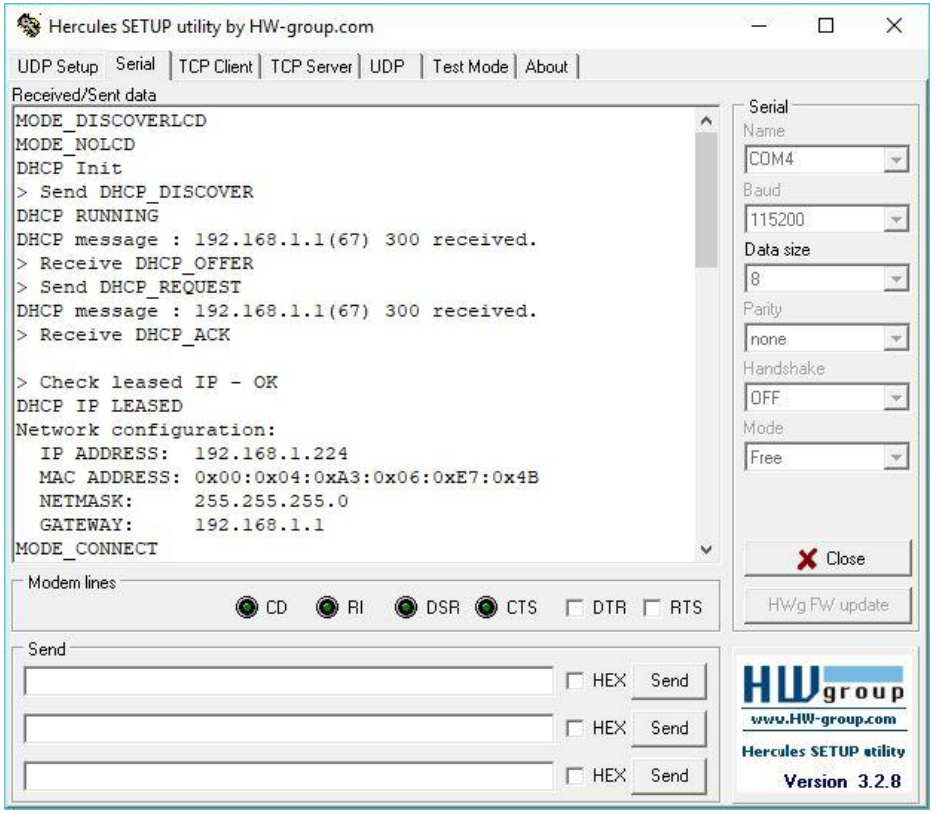
```
case mailto:
  emaildata.emto[0] = 'R';
  emaildata.emto[1] = 'C';
  emaildata.emto[2] = 'P';
  emaildata.emto[3] = 'T';
  emaildata.emto[4] = ' ';
  emaildata.emto[5] = 'T';
  emaildata.emto[6] = 'O';
  emaildata.emto[7] = ':';
  emaildata.emto[8] = '<';
  i = 9;
  j = 6;
  lenmailto = 9;
  do{
        emaildata.emto[i++] = rxBufLCD[j++];
        lenmailto++;
  }while(rxBufLCD[j-1] != 0x00);
  --i;
  emaildata.emto[i++] = '>';
  lenmailto++;
  emaildata.emto[i++] = 0x0D;
  lenmailto++;
  emaildata.emto[i] = 0x0A;
break;
```



Essential Coding Techniques for Hardware Engineers

ARM SMTP Driver: Headless (No LCD) Mode

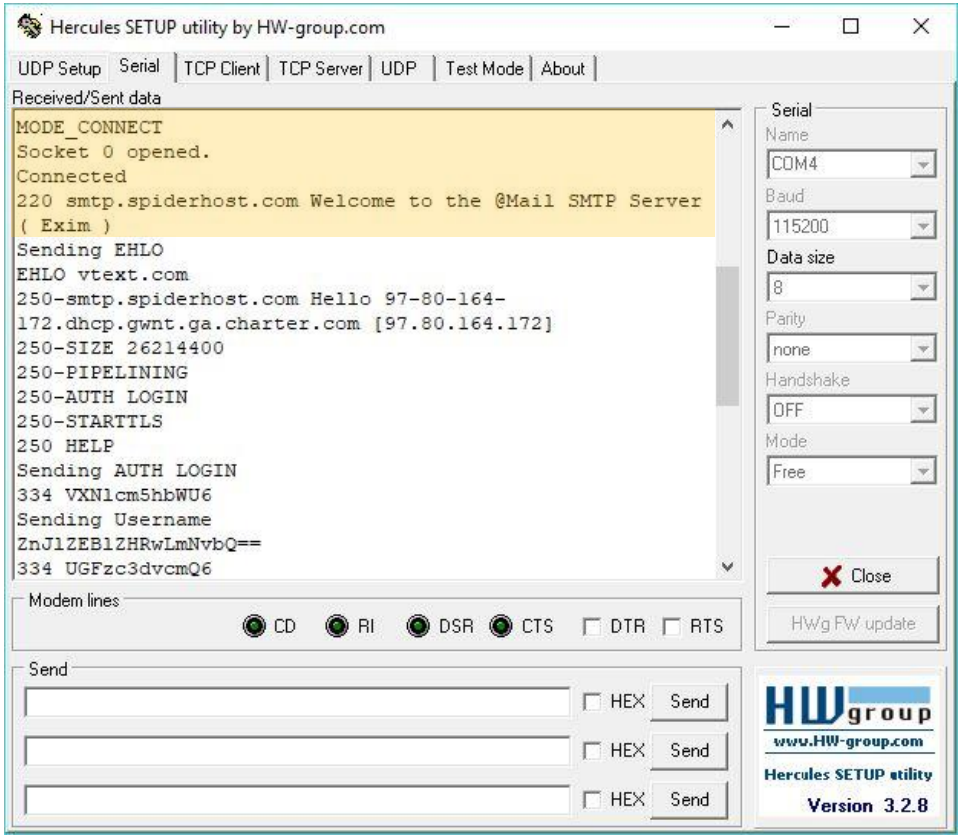
```
case MODE_NOLCD:
  printf("MODE_NOLCD\r\n");
  oneBite = 0x01;
  pageRD(eepageusrnamelen, &lenusrname, &oneBite);
  pageRD(eepagepasswrdlen, &lenpasswrd, &oneBite);
  pageRD(eepagedomainlen, &lenedomain, &oneBite);
  pageRD(eepagemailtolen, &lenmailto, &oneBite);
  pageRD(eepagemailfrmlen, &lenmailfrm, &oneBite);
  pageRD(eepageuname64len, &lenUname64, &oneBite);
  pageRD(eepagepword64len, &lenPword64, &oneBite);

  pageRD(eepageusrname, emaildata.uname, &lenusrname);
  pageRD(eepagepasswrd, emaildata.pword, &lenpasswrd);
  pageRD(eepagedomain, emaildata.domain, &lenedomain);
  pageRD(eepagemailto, emaildata.emto, &lenmailto);
  pageRD(eepagemailfrm, emaildata.emfrm, &lenmailfrm);
  pageRD(eepageuname64, emaildata.uname64, &lenUname64);
  pageRD(eepagepword64, emaildata.pword64, &lenPword64);
  lastpstate = pstate;
  pstate = MODE_DHCP;
break;
```



Essential Coding Techniques for Hardware Engineers

ARM SMTP Driver: Connect to SMTP Server

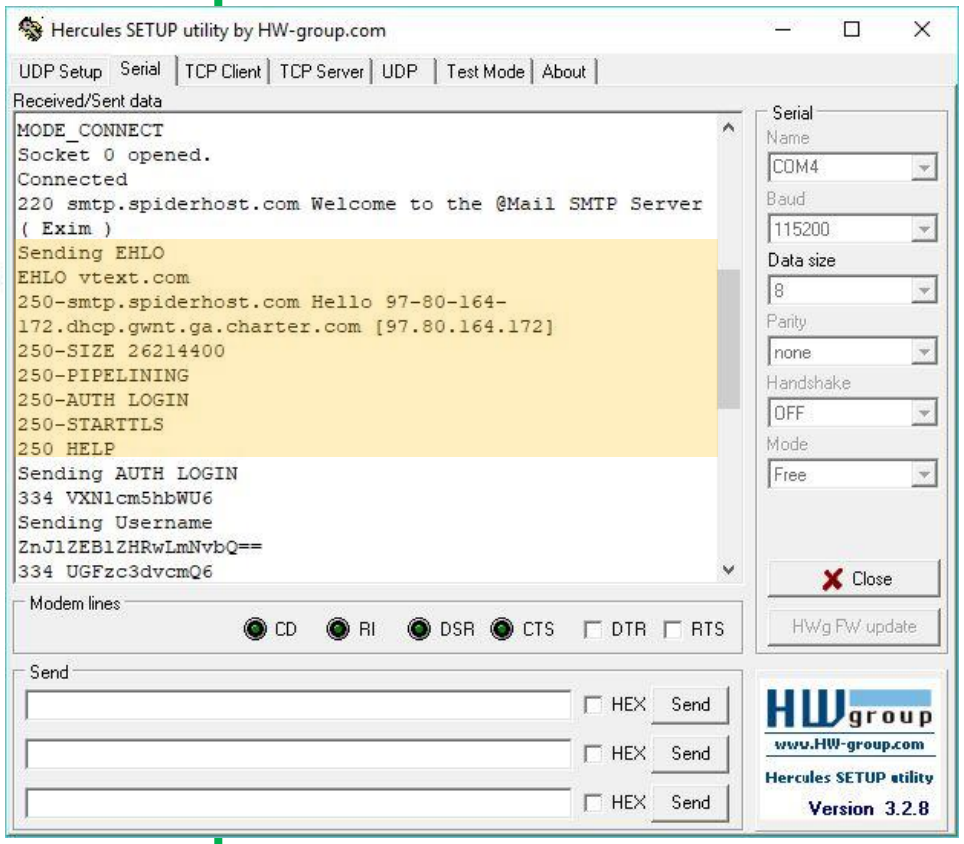
```
case MODE_CONNECT:
  printf("MODE_CONNECT\r\n");
  retVal = socket(0, Sn_MR_TCP, 5000, 0);
  if(retVal == 0)
  {
    printf("Socket 0 opened.\r\n");
    retVal = connect(0, spiderIP, 587);
    if(retVal == SOCK_OK)
    {
      printf("Connected\r\n");
      HAL_Delay(2000);
      rcvLen = 0;
      do{
        rcvLen = getSn_RX_RSR(0);
      }while(rcvLen == 0);
      recv(0, sktRxBuf, rcvLen);
      HAL_UART_Transmit(&huart2, sktRxBuf, rcvLen, 0xFFFF);

      sktIndx = 0;
      do{
      if(sktRxBuf[sktIndx]   == '2' &&
         sktRxBuf[sktIndx+1] == '2' &&
         sktRxBuf[sktIndx+2] == '0' &&
         sktRxBuf[sktIndx+3] == ' ')
        {
          lastpstate = pstate;
          pstate = MODE_EHLO;
        }
      }while(pstate == MODE_CONNECT || sktIndx > rcvLen);
      if(sktIndx > rcvLen)
      {
        lastpstate = pstate;
        pstate = MODE_MONITOR;
        close(0);
      }
    }
  }
  break;
```



Essential Coding Techniques for Hardware Engineers

ARM SMTP Driver: Identify to the SMTP Server

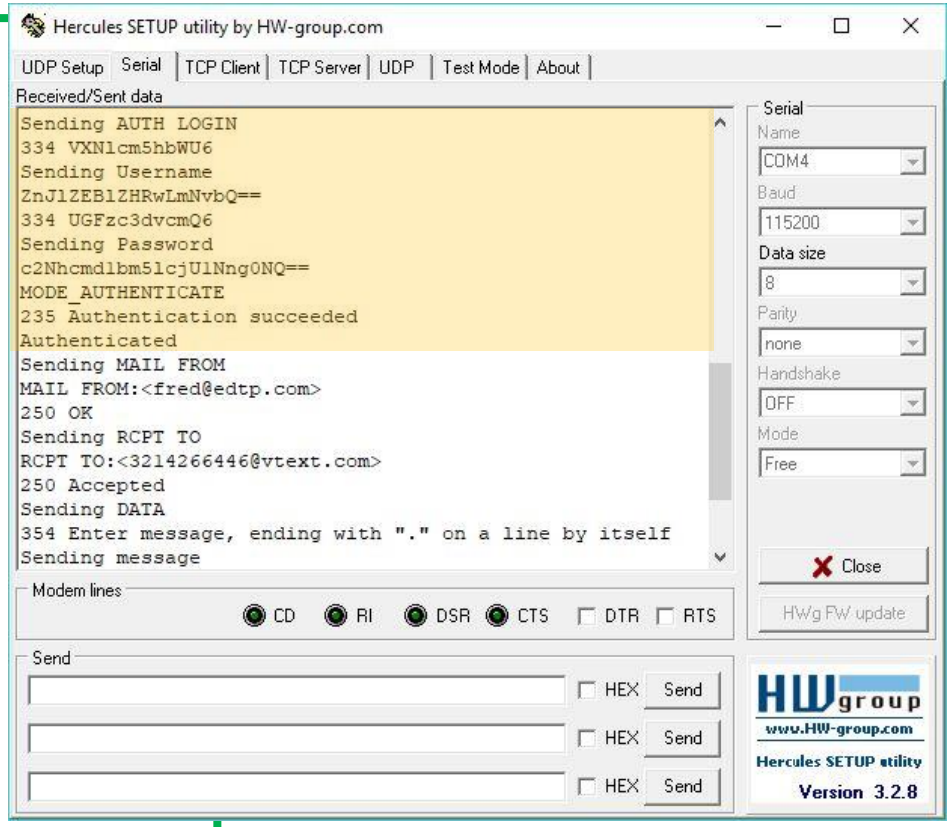
```
case MODE_EHLO:
  printf("Sending EHLO\r\n");
  HAL_UART_Transmit(&huart2, emaildata.domain, lenedomain, 0xFFFF);

  send(0, emaildata.domain, lenedomain);
  HAL_Delay(2000);
  rcvLen = 0;
  do{
    rcvLen = getSn_RX_RSR(0);
  }while(rcvLen == 0);

  recv(0, sktRxBuf, rcvLen);
  HAL_UART_Transmit(&huart2, sktRxBuf, rcvLen, 0xFFFF);

  sktIndx = 0;
  do{
    if(sktRxBuf[sktIndx]   == '2' &&
       sktRxBuf[sktIndx+1] == '5' &&
       sktRxBuf[sktIndx+2] == '0' &&
       sktRxBuf[sktIndx+3] == ' ')
    {
      lastpstate = pstate;
      pstate = MODE_LOGIN;
    }
    sktIndx++;
  }while(pstate == MODE_EHLO || sktIndx > rcvLen);
  if(sktIndx > rcvLen)
  {
    lastpstate = pstate;
    pstate = MODE_MONITOR;
    close(0);
  }
break;
```



Essential Coding Techniques for Hardware Engineers

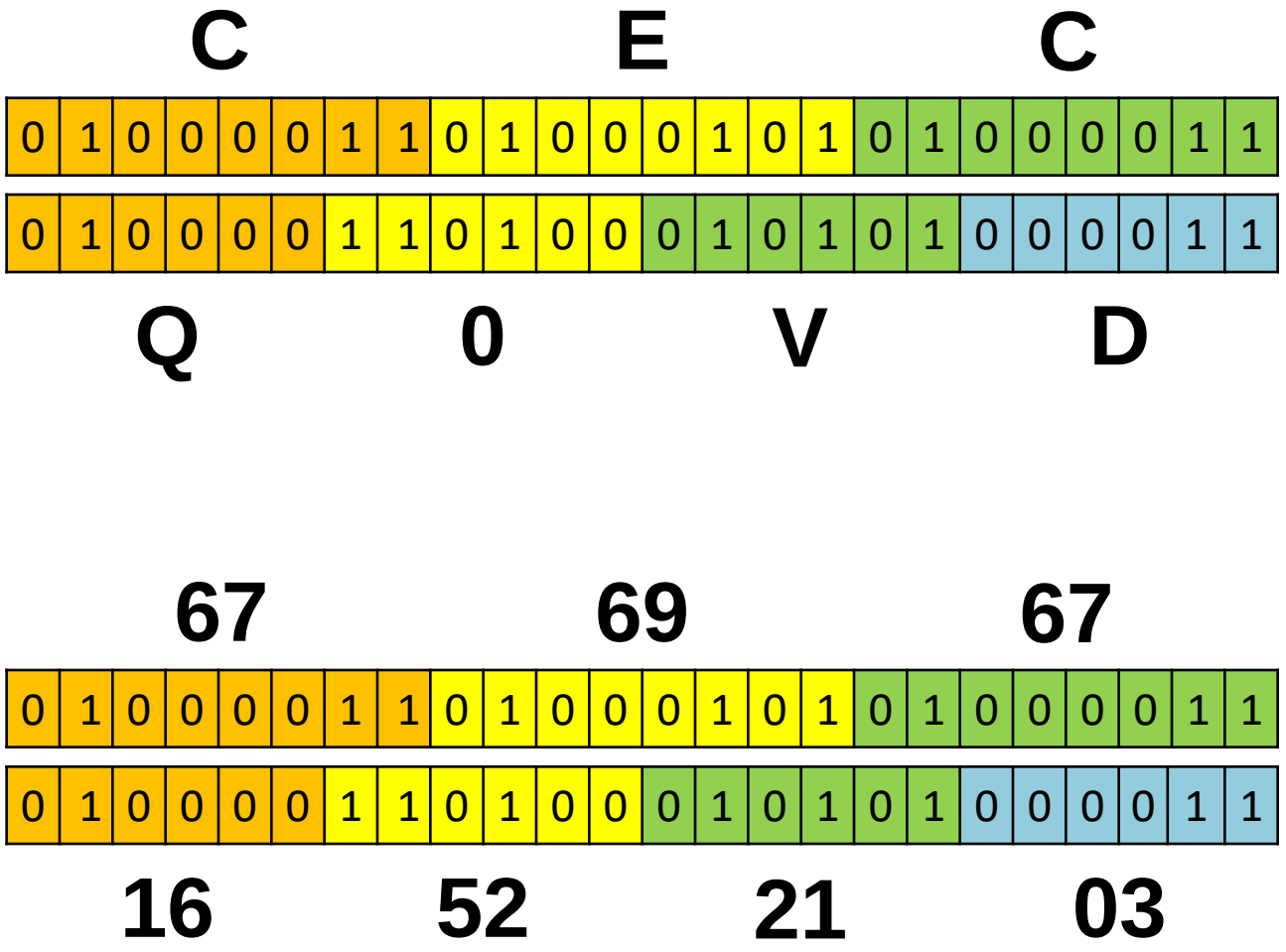
ARM SMTP Driver: LOGIN to the SMTP Server

```
case MODE_LOGIN:
  printf("Sending AUTH LOGIN\r\n");
  sprintf((char*)sktTxBuf, "AUTH LOGIN\r\n");
  send(0,sktTxBuf,strlen((char*)sktTxBuf));
  HAL_Delay(2000);
  rcvLen = 0;
  do{
    rcvLen = getSn_RX_RSR(0);
  }while(rcvLen == 0);

  recv(0, sktRxBuf, rcvLen);
  HAL_UART_Transmit(&huart2, sktRxBuf, rcvLen, 0xFFFF);

  memset(src64, 0x00, sizeof(src64));
  memset(dst64, 0x00, sizeof(dst64));
  sktIndx = 0;
  srcIndx = 0;
  dstIndx = 0;

    if(sktRxBuf[sktIndx++] == '3' &&
       sktRxBuf[sktIndx++] == '3' &&
       sktRxBuf[sktIndx++] == '4' &&
       sktRxBuf[sktIndx++] == ' ')
    {
      do{
        if(sktRxBuf[sktIndx] != 0x0D && sktRxBuf[sktIndx] != 0x0A)
        {
          src64[srcIndx] = sktRxBuf[sktIndx];
          ++srcIndx;
          ++sktIndx;
        }
      }while(sktRxBuf[sktIndx] != 0x0D && sktRxBuf[sktIndx] != 0x0A);
```



Essential Coding Techniques for Hardware Engineers

ARM SMTP Driver: Base64



Essential Coding Techniques for Hardware Engineers

ARM SMTP Driver: Base64 Encode 24 Bits

```
173  static const unsigned char base64_enc_map[64] =
174  {
175      'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J',
176      'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T',
177      'U', 'V', 'W', 'X', 'Y', 'Z', 'a', 'b', 'c', 'd',
178      'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n',
179      'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x',
180      'y', 'z', '0', '1', '2', '3', '4', '5', '6', '7',
181      '8', '9', '+', '/'
182  };
```

67								69								67							
0	1	0	0	0	0	1	1	0	1	0	0	0	1	0	1	0	1	0	0	0	0	1	1
0	1	0	0	0	0	1	1	0	1	0	0	0	1	0	1	0	1	0	0	0	0	1	1
Q								0								V							D
16								52								21							03

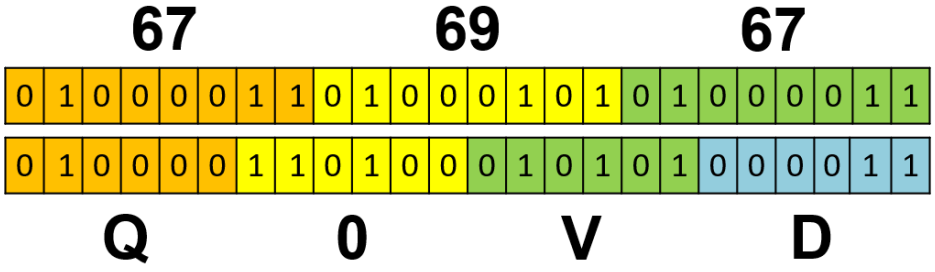
Essential Coding Techniques for Hardware Engineers

ARM SMTP Driver: Base64 Decode 24 Bits

```
//base64_dec_map = ASCII chart used as an index into base64_enc_map
static const unsigned char base64_dec_map[128] =
{
    127, 127, 127, 127, 127, 127, 127, 127, 127, 127,
    127, 127, 127, 127, 127, 127, 127, 127, 127, 127,
    127, 127, 127, 127, 127, 127, 127, 127, 127, 127,
    127, 127, 127, 127, 127, 127, 127, 127, 127, 127,
    127, 127, 127,  62, 127, 127, 127,  63,  52,  53,
     54,  55,  56,  57,  58,  59,  60,  61, 127, 127,
    127,  64, 127, 127, 127,   0,   1,   2,   3,   4,
      5,   6,   7,   8,   9,  10,  11,  12,  13,  14,
     15,  16,  17,  18,  19,  20,  21,  22,  23,  24,
     25, 127, 127, 127, 127, 127, 127,  26,  27,  28,
     29,  30,  31,  32,  33,  34,  35,  36,  37,  38,
     39,  40,  41,  42,  43,  44,  45,  46,  47,  48,
     49,  50,  51, 127, 127, 127, 127, 127
};
```

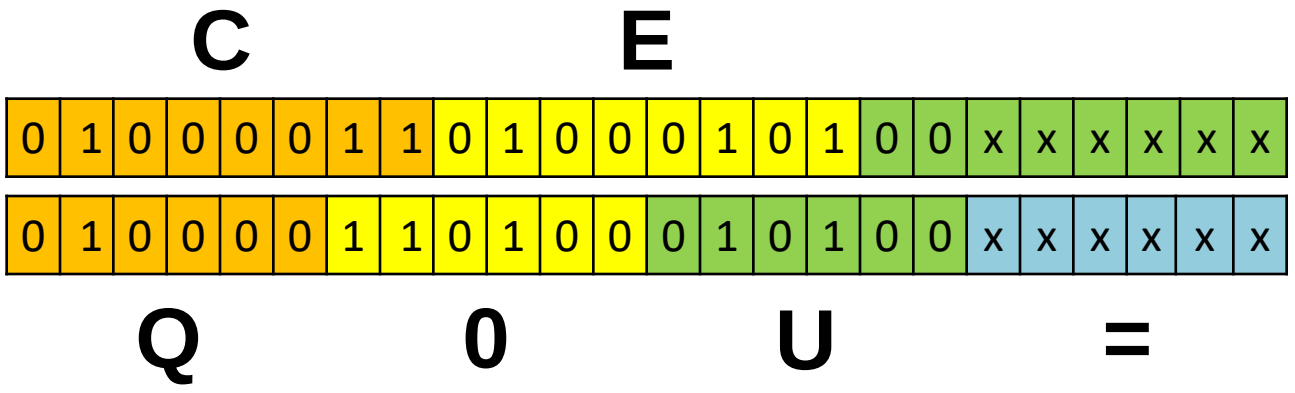
```
173  static const unsigned char base64_enc_map[64] =
174  {
175      'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J',
176      'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T',
177      'U', 'V', 'W', 'X', 'Y', 'Z', 'a', 'b', 'c', 'd',
178      'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n',
179      'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x',
180      'y', 'z', '0', '1', '2', '3', '4', '5', '6', '7',
181      '8', '9', '+', '/'
182  };
```

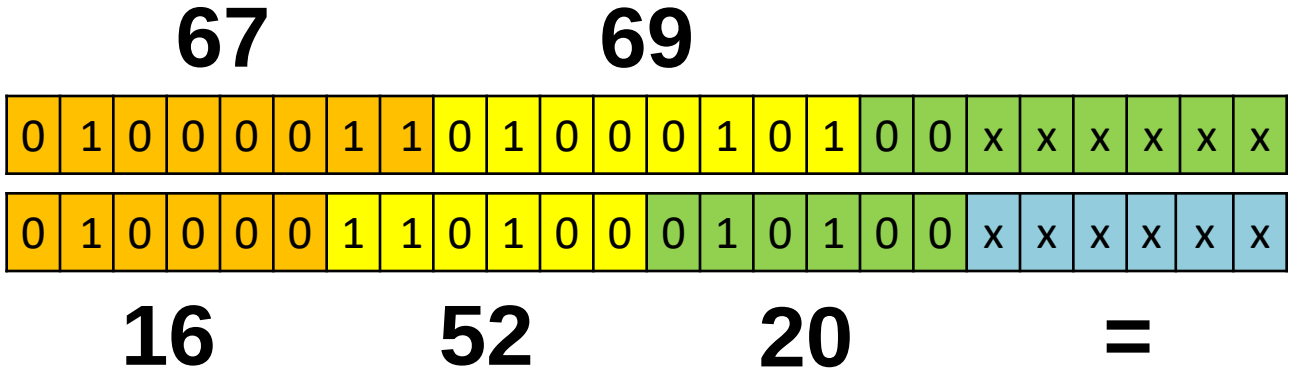
```
//static const unsigned char base64_dec_map[128] =
//{
//    127, 127, 127, 127, 127, 127, 127, 127, 127, 127,
//    127, 127, 127, 127, 127, 127, 127, 127, 127, 127,
//    127, 127, 127, 127, 127, 127, 127, 127, 127, 127,
//    127, 127, 127, 127, 127, 127, 127, 127, 127, 127,
//    127, 127, 127,   +, 127, 127, 127,   /,   0,   1,
//      2,   3,   4,   5,   6,   7,   8,   9, 127, 127,
//    127,   =, 127, 127, 127,   A,   B,   C,   D,   E,
//      F,   G,   H,   I,   J,   K,   L,   M,   N,   O,
//      P,   Q,   R,   S,   T,   U,   V,   W,   X,   Y,
//      Z, 127, 127, 127, 127, 127, 127,   a,   b,   c,
//      d,   e,   f,   g,   h,   i,   j,   k,   l,   m,
//      n,   o,   p,   q,   r,   s,   t,   u,   v,   w,
//      x,   y,   z, 127, 127, 127, 127, 127
//};
```



Essential Coding Techniques for Hardware Engineers

ARM SMTP Driver: Base64 Encode 16 bits





Essential Coding Techniques for Hardware Engineers

ARM SMTP Driver: Base64 Encode 8 bits

C

0	1	0	0	0	0	1	1	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x
0	1	0	0	0	0	1	1	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x

Q w = =

67

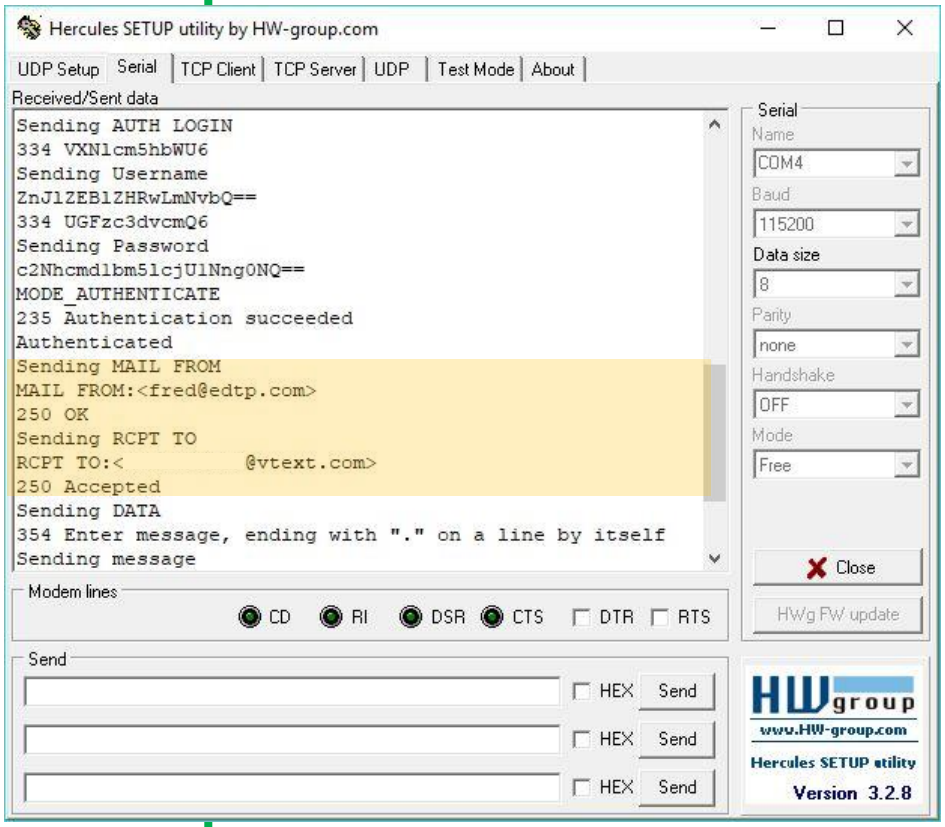
0	1	0	0	0	0	1	1	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x
0	1	0	0	0	0	1	1	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x

16 48 = =

Essential Coding Techniques for Hardware Engineers

ARM SMTP Driver: Send MAIL FROM Data

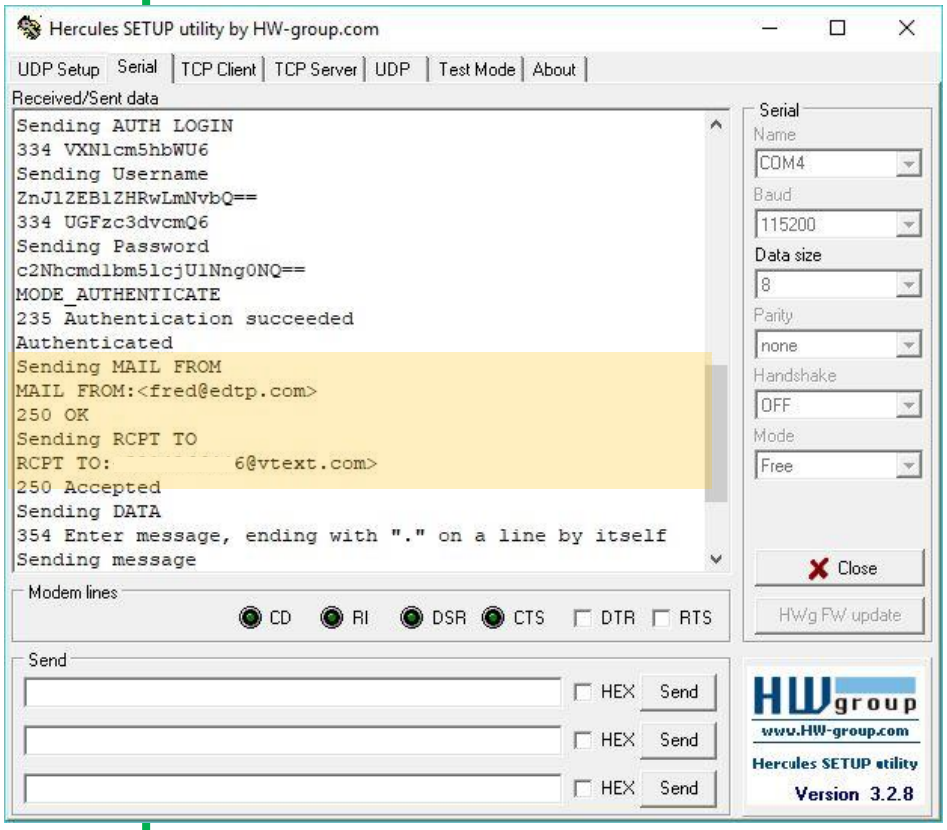
```
case MODE_MAILFROM:
  printf("Sending MAIL FROM\r\n");
  HAL_UART_Transmit(&huart2, emaildata.emfrm, lenmailfrm, 0xFFFF);
  send(0,emaildata.emfrm, lenmailfrm);
  HAL_Delay(2000);
  rcvLen = 0;
  do{
    rcvLen = getSn_RX_RSR(0);
  }while(rcvLen == 0);

  recv(0, sktRxBuf, rcvLen);
  HAL_UART_Transmit(&huart2, sktRxBuf, rcvLen, 0xFFFF);

  sktIndx = 0;

    if(sktRxBuf[sktIndx++] == '2' &&
       sktRxBuf[sktIndx++] == '5' &&
       sktRxBuf[sktIndx++] == '0' &&
       sktRxBuf[sktIndx++] == ' ')
    {
      lastpstate = MODE_MAILFROM;
      pstate = MODE_RCPTTO;
    }
    else
      {
        lastpstate = pstate;
        pstate = MODE_MONITOR;
        close(0);
      }
  break;
```



Essential Coding Techniques for Hardware Engineers

ARM SMTP Driver: Send RCPT TO Data

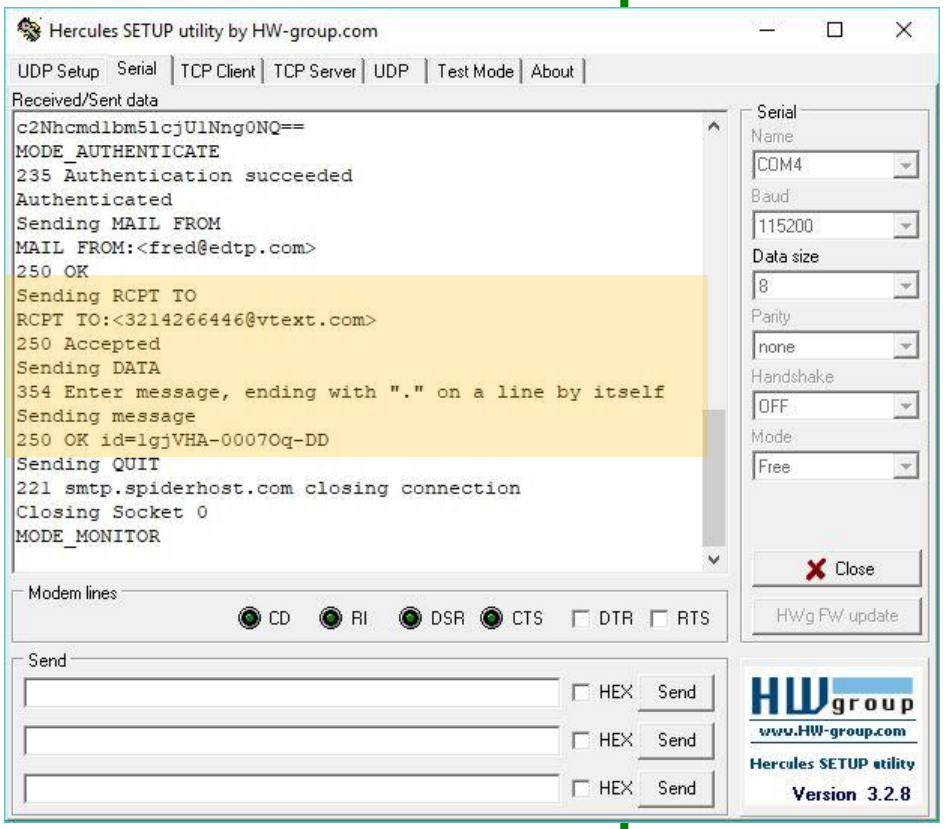
```
case MODE_RCPTTO:
  printf("Sending RCPT TO\r\n");
  HAL_UART_Transmit(&huart2, emaildata.emto, lenmailto, 0xFFFF);
  send(0,emaildata.emto, lenmailto);
  HAL_Delay(2000);
  rcvLen = 0;
  do{
    rcvLen = getSn_RX_RSR(0);
  }while(rcvLen == 0);

  recv(0, sktRxBuf, rcvLen);
  HAL_UART_Transmit(&huart2, sktRxBuf, rcvLen, 0xFFFF);

  sktIndx = 0;

    if(sktRxBuf[sktIndx++] == '2' &&
       sktRxBuf[sktIndx++] == '5' &&
       sktRxBuf[sktIndx++] == '0' &&
       sktRxBuf[sktIndx++] == ' ')
    {
      lastpstate = pstate;
      pstate = MODE_SENDDATA;
    }
    else
      {
        lastpstate = pstate;
        pstate = MODE_MONITOR;
        close(0);
      }
    break;
```



Essential Coding Techniques for Hardware Engineers

ARM SMTP Driver: Send Message

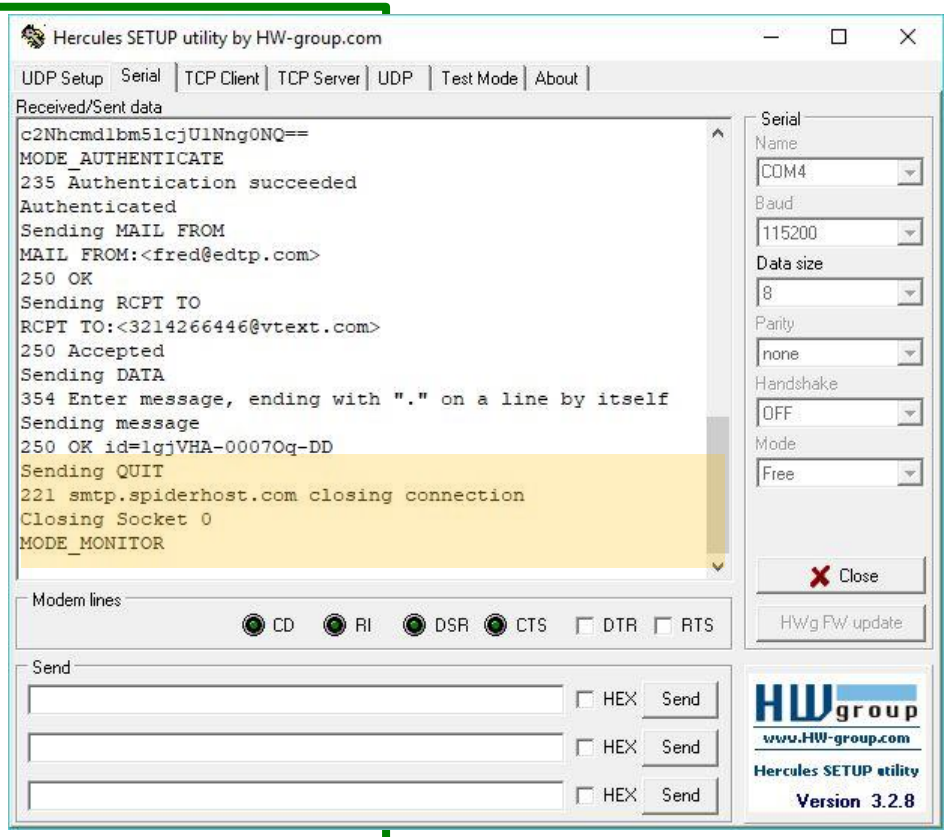
```
case MODE_SENDMAIL:
  printf("Sending message\r\n");
  sprintf((char*)sktTxBuf, "From: Sump Pump\r\nSubject: Moisture Alert\r\nTo: Pump Owner\r\n");
  send(0,sktTxBuf,strlen((char*)sktTxBuf));
  sprintf((char*)sktTxBuf, "Moisture Sensor indicates pump is not running.\r\n.\r\n");
  send(0,sktTxBuf,strlen((char*)sktTxBuf));
  HAL_Delay(2000);
  rcvLen = 0;
  do{
    rcvLen = getSn_RX_RSR(0);
  }while(rcvLen == 0);

  recv(0, sktRxBuf, rcvLen);
  HAL_UART_Transmit(&huart2, sktRxBuf, rcvLen, 0xFFFF);

  sktIndx = 0;

    if(sktRxBuf[sktIndx++] == '2' &&
       sktRxBuf[sktIndx++] == '5' &&
       sktRxBuf[sktIndx++] == '0' &&
       sktRxBuf[sktIndx++] == ' ')
    {
      lastpstate = MODE_SENDDATA;
      pstate = MODE_QUIT;
    }
    else
      {
        lastpstate = pstate;
        pstate = MODE_MONITOR;
        close(0);
      }
  break;
```



Essential Coding Techniques for Hardware Engineers

ARM SMTP Driver: Send Quit and Exit to MONITOR MODE

```
case MODE_QUIT:
      printf("Sending QUIT\r\n");
      sprintf((char*)sktTxBuf, "QUIT\r\n");
      send(0,sktTxBuf,strlen((char*)sktTxBuf));

  HAL_Delay(2000);
  rcvLen = 0;
  do{
    rcvLen = getSn_RX_RSR(0);
  }while(rcvLen == 0);

  recv(0, sktRxBuf, rcvLen);
  HAL_UART_Transmit(&huart2, sktRxBuf, rcvLen, 0xFFFF);

  sktIndx = 0;

    if(sktRxBuf[sktIndx++] == '2' &&
       sktRxBuf[sktIndx++] == '2' &&
       sktRxBuf[sktIndx++] == '1' &&
       sktRxBuf[sktIndx++] == ' ')
    {
      printf("Closing Socket 0\r\n");
      close(0);
    }
    else
      {
      printf("Did not get expected response from server. Closing Socket 0\r\n");
      close(0);
      }
  lastpstate = MODE_QUIT;
  pstate = MODE_MONITOR;
  scratch8++;
break;
```



Essential Coding Techniques for Hardware Engineers

Day 5 Summary