AS608核心代码

录入指纹函数

as608\_function\_result\_t AS608\_InputFingerprint(as608\_handle\_t \*handle, uint16\_t Page\_ID)

{

  as608\_function\_result\_t err\_code = AS608\_SUCCESS;

  while (AS608\_GetWAK(handle) == 0)

    ;

  err\_code = AS608\_GetImage(handle);

  if(err\_code != AS608\_SUCCESS)

  {

    return err\_code;

  }

  err\_code = AS608\_GenChar(handle, CharBuffer1);

  if(err\_code != AS608\_SUCCESS)

  {

    return err\_code;

  }

  if (AS608\_Search(handle, CharBuffer1, 0, 255) == AS608\_SUCCESS)

  {

    return AS608\_SUCCESS;

  }

  while (AS608\_GetWAK(handle) == 0)

    ;

  err\_code = AS608\_GetImage(handle);

  if(err\_code != AS608\_SUCCESS)

  {

    return err\_code;

  }

  err\_code = AS608\_GenChar(handle, CharBuffer2);

  if(err\_code != AS608\_SUCCESS)

  {

    return err\_code;

  }

  err\_code = AS608\_RegModel(handle);

  if(err\_code != AS608\_SUCCESS)

  {

    return err\_code;

  }

  err\_code = AS608\_StoreChar(handle, CharBuffer2, Page\_ID);

  if(err\_code != AS608\_SUCCESS)

  {

    return err\_code;

  }

  return AS608\_SUCCESS;

}

刷指纹函数

as608\_function\_result\_t AS608\_BrushFingerprint(as608\_handle\_t \*handle)

{

  as608\_function\_result\_t err\_code = AS608\_SUCCESS;

  while (AS608\_GetWAK(handle) == 0)

    ;

  err\_code = AS608\_GetImage(handle);

  if(err\_code != AS608\_SUCCESS)

  {

    return err\_code;

  }

  err\_code = AS608\_GenChar(handle, CharBuffer1);

  if(err\_code != AS608\_SUCCESS)

  {

    return err\_code;

  }

  err\_code = AS608\_Search(handle, CharBuffer1, 0, 255);

  return err\_code;

}

RFID核心代码

PCD请求

int RC522\_PcdRequest(rc522\_handle\_t \*handle, uint8\_t req\_code, uint8\_t \*tag\_type)

{

    char status;

    unsigned int unLen;

    unsigned char ucComMF522Buf[MAXRLEN];

    RC522\_ClearRegisterBitMask(handle, Status2Reg, 0x08); /\*清除MFCypto1On位，该位只能通过软件清除\*/

    RC522\_WriteRegister(handle, BitFramingReg, 0x07);     /\*定义传输的最后一个字节发送7位\*/

    RC522\_SetRegisterBitMask(handle, TxControlReg, 0x03); /\*打开非接触式uart发送器\*/

    ucComMF522Buf[0] = req\_code;

    status = RC522\_PcdCommand(handle, PCD\_TRANSCEIVE, ucComMF522Buf, 1, ucComMF522Buf, &unLen);

    if ((status == MI\_OK) && (unLen == 0x10))

    {

        \*tag\_type = ucComMF522Buf[0];

        \*(tag\_type + 1) = ucComMF522Buf[1];

    }

    else

    {

        status = MI\_ERR;

    }

    return status;

}

防冲撞指令

int RC522\_PcdAnticollsion(rc522\_handle\_t \*handle, uint8\_t \*uid)

{

    char status;

    unsigned char i, snr\_check = 0;

    unsigned int unLen;

    unsigned char ucComMF522Buf[MAXRLEN];

    RC522\_ClearRegisterBitMask(handle, Status2Reg, 0x08);

    RC522\_WriteRegister(handle, BitFramingReg, 0x00);

    RC522\_ClearRegisterBitMask(handle, CollReg, 0x80);

    ucComMF522Buf[0] = PICC\_ANTICOLL1;

    ucComMF522Buf[1] = 0x20;

    status = RC522\_PcdCommand(handle, PCD\_TRANSCEIVE, ucComMF522Buf, 2, ucComMF522Buf, &unLen);

    if (status == MI\_OK)

    {

        for (i = 0; i < 4; i++)

        {

            \*(uid + i) = ucComMF522Buf[i];

            snr\_check ^= ucComMF522Buf[i];

        }

        if (snr\_check != ucComMF522Buf[i])

        {

            status = MI\_ERR;

        }

    }

    RC522\_SetRegisterBitMask(handle, CollReg, 0x80);

    return status;

}

ESP01s核心代码

int ESP\_Run(esp\_handle\_t \*p)

{

  static int status = 0;

  static uint32\_t previousMillis = 0, time = 0;

  uint32\_t currentMillis = 0;

  currentMillis = HAL\_GetTick();

  if(currentMillis - previousMillis <= time)

  {

    return -1;

  }

  previousMillis = currentMillis;

  switch(status)

  {

    case 0:

      ESP\_SET\_RST(p, 0);

      HAL\_Delay(5);

      ESP\_SET\_RST(p, 1);

      time = 2000;

      status ++;

      break;

    case 1:

      if(strstr((char \*)p->rxBufferAddr, "OK") != NULL)

      {

        status ++;

      }

      ESP\_TransmitCommand(p, 1000, "AT\r\n");

      time = 1000;

      break;

    case 2:

      if(strstr((char \*)p->rxBufferAddr, "OK") != NULL)

      {

        status ++;

      }

      ESP\_TransmitCommand(p, 1000, "AT+CWMODE=1\r\n");

      time = 1000;

      break;

    case 3:

      if(strstr((char \*)p->rxBufferAddr, "OK") != NULL)

      {

        status ++;

      }

      ESP\_TransmitCommand(p, 1000, "AT+CWDHCP=1,1\r\n");

      time = 1000;

      break;

    case 4:

      if(strstr((char \*)p->rxBufferAddr, "OK") != NULL)

      {

        status ++;

      }

      ESP\_TransmitCommand(p, 1000, "AT+CWJAP=\"%s\",\"%s\"\r\n", p->ssid, p->pwd);

      time = 5000;

      break;

    case 5:

      time = 0;

      p->status = 1;

      break;

    default: break;

  }

  return status;

}

云平台核心代码

int OneNet\_Run(onenet\_handle\_t \*p)

{

  static int status = 0;

  static uint32\_t previousMillis = 0, time = 0;

  uint32\_t currentMillis = 0;

  currentMillis = HAL\_GetTick();

  if(currentMillis - previousMillis <= time || p->esp->status == 0)

  {

    return -1;

  }

  previousMillis = currentMillis;

  switch(status)

  {

    case 0:

      if(strstr((char \*)p->esp->rxBufferAddr, "AT+MQTTUSERCFG=0,1") != NULL)

      {

        status ++;

      }

      else if(strstr((char \*)p->esp->rxBufferAddr, "OK") != NULL)

      {

        status ++;

      }

      ESP\_TransmitCommand(p->esp, 1000, "AT+MQTTUSERCFG=0,1,\"%s\",\"%s\",\"%s\",0,0,\"\"\r\n", p->deviceID, p->productID, p->key);

      time = 4000;

      break;

    case 1:

      if(strstr((char \*)p->esp->rxBufferAddr, "OK") != NULL)

      {

        status ++;

      }

      ESP\_TransmitCommand(p->esp, 1000, "AT+MQTTCONN=0,\"%s\",1883,1\r\n", ONENET\_MQTT\_URL);

      time = 5000;

      break;

    case 2:

      if(strstr((char \*)p->esp->rxBufferAddr, "OK") != NULL)

      {

        status ++;

      }

      ESP\_TransmitCommand(p->esp, 1000, "AT+MQTTSUB=0,\"%s\",1\r\n", p->subscribeTopic[0]);

      time = 3000;

      break;

    case 3:

      if(strstr((char \*)p->esp->rxBufferAddr, "OK") != NULL)

      {

        status ++;

      }

      ESP\_TransmitCommand(p->esp, 1000, "AT+MQTTSUB=0,\"%s\",2\r\n", p->subscribeTopic[1]);

      time = 3000;

      break;

    case 4:

      time = 0;

      if(p->esp->huartx->RxState == HAL\_UART\_STATE\_READY)

      {

        HAL\_UART\_DMAStop(p->esp->huartx);

        memset(p->esp->rxBufferAddr, '\0', p->esp->rxBufferSize);

        HAL\_UARTEx\_ReceiveToIdle\_DMA(p->esp->huartx, p->esp->rxBufferAddr, p->esp->rxBufferSize);

        \_\_HAL\_DMA\_DISABLE\_IT(p->esp->huartx->hdmarx, DMA\_IT\_HT);

      }

      p->externTask(p);

      break;

    default:

      break;

  }

  return status;

}

发送MQTT格式消息

int OneNet\_PostMQTTMessage(onenet\_handle\_t \*p)

{

  int funcResult = 0;

  OneNet\_ChangeStringFormat(p);

  funcResult = ESP\_TransmitCommand(p->esp, 1000, "AT+MQTTPUB=0,\"%s\",\"%s\",0,0\r\n", p->postTopic, p->message);

  /\*---测试函数, 不用可以注释掉---\*/

  // char buffer[256] = {0};

  // HAL\_UART\_Transmit(&huart1, (uint8\_t \*)buffer, strlen(buffer), HAL\_MAX\_DELAY);

  /\*---测试函数结束---\*/

  return funcResult;

}

OV2640摄像头核心代码

int System\_WatchMonitor(system\_handle\_t \*system)

{

  int retval = 0;

  uint8\_t keynum = 0, fingerFlag = 0;

  uint16\_t pixcnt = 0; // 像素统计

  uint32\_t pix = 0;

  uint16\_t linepix = 0;

  uint16\_t linecnt = 0; // 行数统计

  memset(tx3Buffer, '\0', 256);

  LCD\_WriteCommand(system->lcd, 0x36);

  LCD\_WriteData8(system->lcd, 0xC0);//显示正常

  do{

    keynum = Key\_GetNum(system->key);

    fingerFlag = AS608\_GetWAK(system->as608);

    pix = 0;

    pixcnt = 0;  // 像素计数器清零

    linecnt = 0; // 行统计清零

    while (OV2640\_READ\_VSYNC(system->ov2640) != 0); // 等待帧信号

    LCD\_AddressSet( system->lcd,

                    system->lcd->width / 2 - 64,

                    system->lcd->width / 2 + 64 - 1,

                    system->lcd->height / 2 - 80,

                    system->lcd->height / 2 + 80 - 1);

    LCD\_WriteCommand(system->lcd, 0x2c);

    LCD\_WriteDataStart(system->lcd);

    while (linecnt < 160)

    {

      while (OV2640\_READ\_HREF(system->ov2640) != 0)

      {

        while (OV2640\_READ\_PCLK(system->ov2640) != 0);

        while (OV2640\_READ\_PCLK(system->ov2640) == 0);

        tx3Buffer[linepix ++] = OV2640\_READ\_DATA(system->ov2640);

        while (OV2640\_READ\_PCLK(system->ov2640) != 0);

        while (OV2640\_READ\_PCLK(system->ov2640) == 0);

        tx3Buffer[linepix ++] = OV2640\_READ\_DATA(system->ov2640);

        pix++;

      }

      if(pixcnt != pix)

      {

        for(uint16\_t i = 0; i < 256; i += 2)

        {

          LCD\_WriteData16Continue(system->lcd, ((uint16\_t)tx3Buffer[i + 1] << 8) + tx3Buffer[i]);

        }

        pixcnt = pix;

        linepix = 0;

        linecnt ++ ;

      }

    }

    LCD\_WriteDataStop(system->lcd);

  }while(keynum == 0 && fingerFlag == 0);

  LCD\_WriteCommand(system->lcd, 0x36);

  LCD\_WriteData8(system->lcd, 0x00);//显示倒置

  Display\_Clear(system->lcd);

  return 0;

}