Qualcomm Developer Project  
**Gizwits cloud connection**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribution:** |  | | |
| **Email address** | [sunzhen@thundersoft.com](mailto:sunzhen@thundersoft.com)  [yangrong0925@thundersoft.com](mailto:yangrong0925@thundersoft.com)  [wangjie0508@thundersoft.com](mailto:wangjie0508@thundersoft.com)  [Kouzw0723@thundersoft.com](mailto:Kouzw0723@thundersoft.com)  [yansh0810@thundersoft.com](mailto:yansh0810@thundersoft.com) | | |
| **Project Title**\* | Gizwits cloud connection | | |
| **Description**\*  *High level description of the project* ***(75 words or less)*** | This project is designed to use the Gokit4 development kit to send local data  to Gizwits cloud with NB network. | | |
| **Images**  *Upload up to 5 images of your project*  *Please submit/send the original JPEG/PNG files for all images included in the document* | **Gokit4\_board\_619x599.png**  Gokit4_board_619x599  [alt tag: “**Gizwits cloud connection using the Gokit4 development board.**”]  **Usb\_data\_line\_693x271.png**  **Usb_data_line_693x271**  [alt tag: “**using the data line to supply power for Gokit4 development board.** ”]  **data\_in\_Gizwits\_cloud\_676\_404.png**  **data_in_Gizwits_cloud_676_404**  [alt tag: “**for showing the data from local was sent to Gizwits cloud.**”] | | |
| **Objective**   * *What inspired you to create this project?* * *What is your desired outcome?* | The demo named “Gizwits cloud connection” mainly used to show data from local was sent to Gizwits cloud by NB network. | | |
| **Operation System**\*  (Android, Linux, Windows 10 IoT Core) | Android   Linux  RTOS | | Windows 10 IoT Core   Ubuntu Core |
| **Cloud Services/Platform**  AT&T M2X, AWS IoT, IBM Bluemix, IBM Watson IoT, Such as Microsoft Azure IoT) | Amazon AWS IoT   AT&T M2x   IBM Bluemix | | IBM Watson IoT  Gizwits Cloud  Google Cloud Platform |
| **Skill Level Required**  (Beginner, Intermediate, Advanced) | Advanced   Beginner | | Intermediate |
| **Areas of Focus**  (e.g., IoT, smart cities, smart home, robotics, hardware, gaming, healthcare, automotive, digital signage, etc.) | 3D Printing & Modeling   Alexa Voice Service   Bluetooth   Computer Vision   Digital Signage   Education   Embedded    Gaming | | Healthcare   NB-iot   Robotics   Sensors   Smart Cities   Toys  Low Power |
| **Materials Required / Parts List / Tools** | Part Name | Link | |
| BG96 | <https://www.quectel.com/product/bg96.htm> | |
| Gokit4 | <https://www.csdn.net/article/a/2018-09-03/15959591> | |
| **Source Code / Source Examples / Application Executable**  *Link to open source / shareable code repository* | Description | Link | |
| [Source Code](https://github.com/canyudeguang/Home_Automation) | <https://github.com/ThunderSoft-XA/demo-Gizwits-cloud-connection> | |
| **Additional Resources**  *List related links or resources such as websites, videos, presentations, or other materials* | Resource Title | Link or File Name (and provide file) | |
| Because there is no other parts except for a single board, so no video to show the results. | | |
| **Build / Assembly Instructions** | Parts used Below are the items used in this project.  **parts.png**  2018-12-17 15:19:51屏幕截图  [alt tag: “**Parts used for the Gokit4 development kit Gizwits cloud connection**”]   1. Mobile Phone which installed a universal apk provided by Gizwits cloud, used to see data from local. 2. Gokit4 development board. 3. Usb data line. (supply power.) 4. Win7 PC (user can view data from local on Gizwits cloud.)   Deploying the project   1. Define products on the Gizwits cloud platform,including basic information,data point,hardware solution,etc. 2. Generate code based on the soc scheme.(Because the Gizwits cloud does not currently support generating code based on the MDM9206 platform, we need to choose ESP\_826632M at hardware platform,then do some replacement.) 3. Add local data which will be sent to Gizwis cloud. 4. Register moblie IOT card to NB network. 5. Compile the code and flash the image. 6. If no problem, upload code to Github.   How does it work?  First of all, we must know how data was sent to Gizwits cloud.  Reporting process:  userTimeCB----->gizwitsHandle----->gizCheckReport---->gizDataPoints2ReportData----->gagentUploadData  demo-Gizwits-cloud-connection/main/main.c  void gagentMain( void )  {  getFreeHeap();  sensorInit();  gizwitsInit();  }  The function named gagentMain was called by GAgent, the main role of GAgent is data forwarding, which is a data interaction bridge between device data, wit cloud, and application end (APP).in function sensorInit, do some sensor init, and there is a function called tx\_timer\_create.  void sensorInit(void)  {  int32 ret = -1;  gizLog(LOG\_INFO,"Sensor initialization ...\n");  led\_init(); //led init  motor\_init(); //motor init  heartrate\_init(); //plusensor init  motion\_init(); //adxl345 init  pressure\_init(); //fsr402 init  txm\_module\_object\_allocate(&userTimer, sizeof(TX\_TIMER));  ret = tx\_timer\_create(userTimer, "userTimer", userTimerCB, NULL, 1,  200, TX\_AUTO\_ACTIVATE);  if(ret != TX\_SUCCESS)  {  gizLog(LOG\_WARNING,"Failed to create UserTimer.\n");  }  }  userTimeCB is registered in tx\_timer\_create, and the callback function is called back at regular intervals.  userTimeCB is used to obtain report data of user area.  void ICACHE\_FLASH\_ATTR userTimerCB(void)  {  gizLog(LOG\_INFO,"in userTimerCB.....\n");  static uint8\_t ctime = 0;  static uint8\_t ccount = 0;  int8\_t status = 0;  // set some value, which will be sent to GIzwits cloud.  uint32\_t pressure = 80; //pressure  uint32\_t heartrate = 75; //heartbeat  int32\_t X\_axis\_Value = 50; //x axis value  int32\_t Y\_axis\_Value = 60; //y axis value  int32\_t Z\_axis\_Value = 70; //z axis value    if (QUERY\_INTERVAL < ctime)  {  ctime = 0; //come in per 5 times  //status = getHeartRateValue(&heartrate); //heartrate from Pulsesensor  if( status )  {  gizLog(LOG\_INFO,"get health info error\n");    }  currentDataPoint.valueHeartRateValue = heartrate;  if(currentDataPoint.valueHeartRateValue > HEART\_RATE\_THRESHOLD) {  //set\_gpio\_value(0x00, led\_pin\_num\_Pulsesensor);  }  //status = getaxis(&X\_axis\_Value,&Y\_axis\_Value,&Z\_axis\_Value); //axis from ADXL345  if( status )  {  gizLog(LOG\_INFO,"get stepcount error\n");    }  currentDataPoint.valueX\_axis\_Value = X\_axis\_Value;  currentDataPoint.valueY\_axis\_Value = Y\_axis\_Value;  currentDataPoint.valueZ\_axis\_Value = Z\_axis\_Value;  if(currentDataPoint.valueX\_axis\_Value > X\_AXIS\_THRESHOLD || currentDataPoint.valueY\_axis\_Value > Y\_AXIS\_THRESHOLD  || currentDataPoint.valueZ\_axis\_Value > Z\_AXIS\_THRESHOLD)  {  //set\_gpio\_value(0x00, led\_pin\_num\_motion);  }  //status = getPressValue(&pressure); //pressure pressure sensor  if( status )  {  gizLog(LOG\_INFO,"get pressure error\n");  }  currentDataPoint.valuePressure\_Value = pressure;  if (currentDataPoint.valuePressure\_Value >= PRESSURE\_THRESHOLD)  {  set\_gpio\_value(0x00, motor\_pin\_num);  }  gizLog(LOG\_INFO,"begin to upload data\n");  gizwitsHandle((dataPoint\_t \*)&currentDataPoint);  ccount++; //record collect times  gizLog(LOG\_INFO,"ccount = %d\n", ccount);  }  ctime++;  }  gizwitsHandle is used to complete the change of device data.  demo-Gizwits-cloud-connection/Gizwits/gizwits\_protocol.c  int8\_t ICACHE\_FLASH\_ATTR gizwitsHandle(dataPoint\_t \*dataPoint)  {  if(NULL == dataPoint)  {  gizLog(LOG\_WARNING,"!!! gizReportData Error \n");  return (-1);  }  //Regularly report conditional  if((1 == gizCheckReport(dataPoint, (dataPoint\_t \*)&gizwitsProtocol.gizLastDataPoint)))  {  gizDataPoints2ReportData(dataPoint, &gizwitsProtocol.reportData.devStatus);  gizwitsProtocol.reportData.action = ACTION\_REPORT\_DEV\_STATUS;  gagentUploadData(NULL, (uint8\_t \*)&gizwitsProtocol.reportData, sizeof(gizwitsReport\_t),getConnectM2MStatus(),gizwitsProtocol.mac, uploadDataCBFunc);  gizLog(LOG\_INFO,"~~~reportData \n");  //printf\_bufs((uint8\_t \*)&gizwitsProtocol.reportData, sizeof(gizwitsReport\_t));  gizMemcpy((uint8\_t \*)&gizwitsProtocol.gizLastDataPoint, (uint8\_t \*)dataPoint, sizeof(dataPoint\_t));  }  return 0;  }  gizCheckReport is used to determine whether to report the current status data.  static int8\_t ICACHE\_FLASH\_ATTR gizCheckReport(dataPoint\_t \*cur, dataPoint\_t \*last)  {  int8\_t ret = 0;  static uint32\_t lastReportTime = 0;  uint32\_t currentTime = 0;  if((NULL == cur) || (NULL == last))  {  gizLog(LOG\_WARNING,"gizCheckReport Error , Illegal Param\n");  return -1;  }  currentTime = gizGetTimerCount();  if(last->valueLedValue != cur->valueLedValue)  {  gizLog(LOG\_INFO,"valueLedValue Changed\n");  ret = 1;  }  .......  if(last->valuePressure\_Value != cur->valuePressure\_Value)  {  if(currentTime - lastReportTime >= REPORT\_TIME\_MAX)  {  gizLog(LOG\_INFO, "valuePressure\_Value Changed\n");  ret = 1;  }  }  if(1 == ret)  {  lastReportTime = gizGetTimerCount();  }  return ret;  }  gizDataPoints2ReportData is used to complete conversion of user area data to report type data.  static int8\_t ICACHE\_FLASH\_ATTR gizDataPoints2ReportData(dataPoint\_t \*dataPoints , devStatus\_t \*devStatusPtr)  {  if((NULL == dataPoints) || (NULL == devStatusPtr))  {  gizLog(LOG\_WARNING,"gizDataPoints2ReportData Error , Illegal Param\n");  return -1;  }  gizMemset((uint8\_t \*)devStatusPtr->wBitBuf,0,sizeof(devStatusPtr->wBitBuf));  gizStandardCompressValue(LedValue\_BYTEOFFSET,LedValue\_BITOFFSET,LedValue\_LEN,(uint8\_t \*)devStatusPtr,dataPoints->valueLedValue);  gizStandardCompressValue(MotorValue\_BYTEOFFSET,MotorValue\_BITOFFSET,MotorValue\_LEN,(uint8\_t \*)devStatusPtr,dataPoints->valueMotorValue);  gizByteOrderExchange((uint8\_t \*)devStatusPtr->wBitBuf,sizeof(devStatusPtr->wBitBuf));  devStatusPtr->valueHeartRateValue = gizY2X(HeartRateValue\_RATIO, HeartRateValue\_ADDITION, dataPoints->valueHeartRateValue);  devStatusPtr->valueX\_axis\_Value = exchangeBytes(gizY2X(X\_axis\_Value\_RATIO, X\_axis\_Value\_ADDITION, dataPoints->valueX\_axis\_Value));  devStatusPtr->valueY\_axis\_Value = exchangeBytes(gizY2X(Y\_axis\_Value\_RATIO, Y\_axis\_Value\_ADDITION, dataPoints->valueY\_axis\_Value));  devStatusPtr->valueZ\_axis\_Value = exchangeBytes(gizY2X(Z\_axis\_Value\_RATIO, Z\_axis\_Value\_ADDITION, dataPoints->valueZ\_axis\_Value));  devStatusPtr->valuePressure\_Value = exchangeBytes(gizY2X(Pressure\_Value\_RATIO, Pressure\_Value\_ADDITION, dataPoints->valuePressure\_Value));  return 0;  }  gagentUploadData is used to send the reported data to the communication module.  After the above steps, user area data will be uploaded to the Gizwits cloud , below are some usage instructions to test the project. | | |
| **Usage Instructions** | 1. Downloading code from github according to the repository in “<https://github.com/ThunderSoft-XA/demo-Gizwits-cloud-connection>” sheet. 2. Compile the code and flash the image to Gokit4 development kit. 3. Use USB data line to supply power. 4. Click on “Device log” in Gizwits cloud to check if device is online. 5. When device online, we can click “view” button, and we will see data from local was sent to Gizwits cloud. 6. We can also see those data from android phone with a universal apk provided by Gizwits cloud, and it’s easy to operate. | | |
| **Contributor(s) Info**  *Feel free to include headshots!* | Name | Title  Company | |
| Zhen  [sunzhen@thundersoft.com](mailto:yangrong0925@thundersoft.com) | Thundersoft | |
| Rong  [yangrong0925@thundersoft.com](mailto:yangrong0925@thundersoft.com) | Thundersoft | |
| Jie  [wangjie0508@thundersoft.com](mailto:wangjie0508@thundersoft.com) | Thundersoft | |
| Kou  [kouzw0723@thundersoft.com](mailto:Kouzw0723@thundersoft.com) | Thundersoft | |
| Eric  [yansh0810@thundersoft.com](mailto:yansh0810@thundersoft.com) | Thundersoft | |

*By submitting your content (“Submission”), you are granting Qualcomm a royalty-free, perpetual, non-exclusive, unrestricted, worldwide license to: (a) post, use, copy, sublicense, adapt, transmit, publicly perform or display any such Submission, (b) use, reproduce, modify, adapt, publish, translate, create derivative works from, distribute, perform, play, host, communicate, make available and publish your Submission without restriction and (c) sublicense to third parties the unrestricted right to exercise any of the foregoing rights granted with respect to the Submission. The foregoing grants shall include the right to exploit any ideas, concepts, intellectual property, or proprietary rights in such Submission, including but not limited to rights under copyright, trademark, servicemark or patent laws under any relevant jurisdiction without Qualcomm owing any monies to you whatsoever. You represent and warrant that you own all right, title and interest in and to the Submission, or you have been granted sufficient rights in and to the Submission allowing the foregoing use of such Submission.*