Qualcomm Developer Project  
demo-Smart-vibrator

|  |  |  |  |
| --- | --- | --- | --- |
| **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**\* | **demo-Smart-vibrator** | | |
| **Description**\*  *High level description of the project* ***(75 words or less)*** | Mainly to use the APP installed in mobile phone to send commands to control the vibrator provided by Gokit4 development kit to start/stop the vibration. | | |
| **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.** ”]  **led\_348x225.png**   |  | | --- | | **led_348x225** |   [alt tag: “**app can send control commands to on/off led. ”]**  **Data&Comm\_781x397.png**  **Data&Comm**  [alt tag: “**for showing the data from local and commands from apk.**”] | | |
| **Objective**   * *What inspired you to create this project?* * *What is your desired outcome?* | The demo named “demo-Smart-viblitor” mainly to use the APP installed in mobile phone to send commands to control the vibrator provided by Gokit4 development kit to start/ stop the vibration. | | |
| **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-Smart-vibrator> | |
| **Additional Resources**  *List related links or resources such as websites, videos, presentations, or other materials* | Resource Title | Link or File Name (and provide file) | |
| [https://pan.baidu.com/s/12QIOdFPGJ4VtaGiCqQ1aKQ](https://pan.baidu.com/s/12QIOdFPGJ4VtaGiCqQ1aKQ )  提取码：upc1 | | |
| **Build / Assembly Instructions** | Parts used Below are the items used in this project.  **parts.png**   |  | | --- | | vibrator |   [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 and control vibrator provided by Gokit4 development kit. 2. Gokit4 development board. 3. Usb data line. (supply power.) 4. Win7 PC.   Deploying the project   1. Define product named “Smart\_vibrator” 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.replacement can refer to:<http://club.gizwits.com/thread-9395-1-1.html>.) 3. Add local data which will be sent to Gizwis cloud. 4. Install apk from “<https://download.gizwits.com/zh-cn/p/98/99>”. 5. Register moblie IOT card to NB network. 6. Add driver related code。 7. Compile the code and flash the image. 8. Run the device,open APP and control it. 9. If no problem, upload code to Github.   How does it work?  First of all, let’s see gpio and motor init.  demo-Smart-vibrator/main/main.c  void gagentMain( void )  {  getFreeHeap();  sensorInit();  gizwitsInit();  }  void sensorInit(void)  {  int32 ret = -1;    gizLog(LOG\_INFO,"Sensor initialization ...\n");  led\_init();  motor\_init();  txm\_module\_object\_allocate(&userTimer, sizeof(TX\_TIMER));  ret = tx\_timer\_create(userTimer, "userTimer", userTimerCB, NULL, 1,  100, TX\_AUTO\_ACTIVATE);  if(ret != TX\_SUCCESS)  {  gizLog(LOG\_WARNING,"Failed to create UserTimer.\n");  }  }  void led\_init()  {  gizLog(LOG\_INFO,"in led\_init ...\n");  led\_gpio\_config();  }  void motor\_init()  {  gizLog(LOG\_INFO,"in motor\_init ...\n");  motor\_gpio\_config();  }  demo-Smart-vibrator/gpio/gpio.c  void led\_gpio\_config()  {  gizLog(LOG\_INFO,"in led\_gpio\_config...\n");    gpio\_config(GPIO\_BLUE, QAPI\_GPIO\_OUTPUT\_E, QAPI\_GPIO\_NO\_PULL\_E, QAPI\_GPIO\_2MA\_E);    gpio\_config(GPIO\_RED, QAPI\_GPIO\_OUTPUT\_E, QAPI\_GPIO\_NO\_PULL\_E, QAPI\_GPIO\_2MA\_E);    gpio\_config(GPIO\_GREEN, QAPI\_GPIO\_OUTPUT\_E, QAPI\_GPIO\_NO\_PULL\_E, QAPI\_GPIO\_2MA\_E);    }  void motor\_gpio\_config()  {  gizLog(LOG\_INFO,"in motor\_gpio\_config ...\n");    gpio\_config(GPIO\_MOTOR, QAPI\_GPIO\_OUTPUT\_E, QAPI\_GPIO\_NO\_PULL\_E, QAPI\_GPIO\_2MA\_E);  }  gpio\_config: do some gpio initialization  void gpio\_config(MODULE\_PIN\_ENUM m\_pin,qapi\_GPIO\_Direction\_t gpio\_dir,qapi\_GPIO\_Pull\_t gpio\_pull,qapi\_GPIO\_Drive\_t gpio\_drive)  {  gizLog(LOG\_INFO,"in gpio config.....\n");    qapi\_Status\_t status = QAPI\_OK;  tlmm\_config[m\_pin].pin = gpio\_map\_tbl[m\_pin].gpio\_id;  tlmm\_config[m\_pin].func = gpio\_map\_tbl[m\_pin].gpio\_func;  tlmm\_config[m\_pin].dir = gpio\_dir;  tlmm\_config[m\_pin].pull = gpio\_pull;  tlmm\_config[m\_pin].drive = gpio\_drive;  // the default here  status = qapi\_TLMM\_Get\_Gpio\_ID(&tlmm\_config[m\_pin], &gpio\_id\_tbl[m\_pin]);    gizLog(LOG\_INFO,"pin\_num = %d, gpio\_id[%d], status = %d ...\n",gpio\_map\_tbl[m\_pin].pin\_num, gpio\_map\_tbl[m\_pin].gpio\_id, status);    if (status == QAPI\_OK)  {  status = qapi\_TLMM\_Config\_Gpio(gpio\_id\_tbl[m\_pin], &tlmm\_config[m\_pin]);  gizLog(LOG\_INFO,"after qapi\_TLMM\_Config\_Gpio, status = %d ...\n", status);    if (status != QAPI\_OK)  {  gizLog(LOG\_INFO,"gpio config failed.....\n");  }    status = qapi\_TLMM\_Drive\_Gpio(gpio\_id\_tbl[m\_pin], gpio\_map\_tbl[m\_pin].gpio\_id, QAPI\_GPIO\_HIGH\_VALUE\_E);  }  }  Process：  gizIssuedProcess ---> ACTION\_CONTROL\_DEVICE ---> gizDataPoint2Event ---> gizwitsEventProcess  gizIssuedProcess: the function is called by the gagent to receive the relevant protocol data delivered from the cloud or the app.  int32\_t ICACHE\_FLASH\_ATTR gizIssuedProcess(uint8\_t \*didPtr, uint8\_t \*inData, uint32\_t inLen,uint8\_t \*outData,int32\_t \*outLen)  {  uint8\_t i = 0;  if((NULL == inData) || (NULL == outData) || (NULL == outLen))  {  gizLog(LOG\_WARNING,"!!! IssuedProcess Error \n");  return -1;  }  if(NULL == didPtr)  {  gizLog(LOG\_INFO,"~~~gizIssuedProcess: did is NULL .\n");  }  else  {  gizLog(LOG\_INFO,"~~~gizIssuedProcess: did %s\n", didPtr);  }  gizLog(LOG\_INFO,"%s: ", "~~~issued data");  //printf\_bufs((uint8\_t \*)inData,inLen);  if(NULL == didPtr)  {  switch(inData[0])  {  case ACTION\_CONTROL\_DEVICE:  gizDataPoint2Event((gizwitsIssued\_t \*)&inData[1], &gizwitsProtocol.issuedProcessEvent,&gizwitsProtocol.gizCurrentDataPoint);  gizwitsEventProcess(&gizwitsProtocol.issuedProcessEvent, (uint8\_t \*)&gizwitsProtocol.gizCurrentDataPoint, sizeof(dataPoint\_t));  gizMemset((uint8\_t \*)&gizwitsProtocol.issuedProcessEvent, 0, sizeof(eventInfo\_t));  \*outLen = 0;  break;    case ACTION\_READ\_DEV\_STATUS:  gizDataPoints2ReportData(&gizwitsProtocol.gizLastDataPoint,&gizwitsProtocol.reportData.devStatus);  gizwitsProtocol.reportData.action = ACTION\_READ\_DEV\_STATUS\_ACK;  gizMemcpy(outData, (uint8\_t \*)&gizwitsProtocol.reportData, sizeof(gizwitsReport\_t));  \*outLen = sizeof(gizwitsReport\_t);    gizLog(LOG\_INFO,"%s: ", "~~~ReadReport \n");  //printf\_bufs((uint8\_t \*)outData,\*outLen);  break;    case ACTION\_W2D\_TRANSPARENT\_DATA:  gizMemcpy(gizwitsProtocol.transparentBuff, &inData[1], inLen-1);  gizwitsProtocol.transparentLen = inLen-1;    gizwitsProtocol.issuedProcessEvent.event[0] = TRANSPARENT\_DATA;  gizwitsProtocol.issuedProcessEvent.num = 1;  gizwitsEventProcess(&gizwitsProtocol.issuedProcessEvent, (uint8\_t \*)gizwitsProtocol.transparentBuff, gizwitsProtocol.transparentLen);  gizMemset((uint8\_t \*)&gizwitsProtocol.issuedProcessEvent, 0, sizeof(eventInfo\_t));  gizMemset((uint8\_t \*)gizwitsProtocol.transparentBuff, 0, BUFFER\_LEN\_MAX);  gizwitsProtocol.transparentLen = 0;  \*outLen = 0;  break;    default:  break;  }  }  else  {  gizLog(LOG\_WARNING," Error : didPtr \n");  }  return 0;  }  ACTION\_CONTROL\_DEVICE: Perform related processing of "controlled protocol"  gizDataPoint2Event: Generate "control events" according to the protocol and complete the conversion of the corresponding data type.  static int8\_t ICACHE\_FLASH\_ATTR gizDataPoint2Event(gizwitsIssued\_t \*issuedData, eventInfo\_t \*info, dataPoint\_t \*dataPoints)  {  if((NULL == issuedData) || (NULL == info) ||(NULL == dataPoints))  {  gizLog(LOG\_WARNING,"gizDataPoint2Event Error , Illegal Param\n");  return -1;  }    /\*\* Greater than 1 byte to do bit conversion \*\*/  if(sizeof(issuedData->attrFlags) > 1)  {  if(-1 == gizByteOrderExchange((uint8\_t \*)&issuedData->attrFlags,sizeof(attrFlags\_t)))  {  gizLog(LOG\_WARNING,"gizByteOrderExchange Error\n");  return -1;  }  }  if(0x01 == issuedData->attrFlags.flagLed\_Value)  {  info->event[info->num] = EVENT\_Led\_Value;  info->num++;  dataPoints->valueLed\_Value = gizStandardDecompressionValue(Led\_Value\_BYTEOFFSET,Led\_Value\_BITOFFSET,Led\_Value\_LEN,(uint8\_t \*)&issuedData->attrVals.wBitBuf,sizeof(issuedData->attrVals.wBitBuf));  }    if(0x01 == issuedData->attrFlags.flagMotor\_Value)  {  info->event[info->num] = EVENT\_Motor\_Value;  info->num++;  dataPoints->valueMotor\_Value = gizStandardDecompressionValue(Motor\_Value\_BYTEOFFSET,Motor\_Value\_BITOFFSET,Motor\_Value\_LEN,(uint8\_t \*)&issuedData->attrVals.wBitBuf,sizeof(issuedData->attrVals.wBitBuf));  }  return 0;  }  gizwitsEventProcess: corresponding event processing according to the generated "control event" ( calling the corresponding driver function)  int8\_t ICACHE\_FLASH\_ATTR gizwitsEventProcess(eventInfo\_t \*info, uint8\_t \*data, uint32\_t len)  {  uint8\_t i = 0;  dataPoint\_t \* dataPointPtr = (dataPoint\_t \*)data;  moduleStatusInfo\_t \* wifiData = (moduleStatusInfo\_t \*)data;  if((NULL == info) || (NULL == data))  {  gizLog(LOG\_WARNING,"!!! gizwitsEventProcess Error \n");  return -1;  }  for(i = 0; i < info->num; i++)  {  switch(info->event[i])  {  case EVENT\_Led\_Value :  currentDataPoint.valueLed\_Value = dataPointPtr->valueLed\_Value;  gizLog(LOG\_INFO, "Evt: EVENT\_LedValue %d \n", currentDataPoint.valueLed\_Value);  if(0x01 == currentDataPoint.valueLed\_Value)  {  //on led  gpio\_high\_low(true,GPIO\_GREEN);  }  else  {  //off led  gpio\_high\_low(false, GPIO\_GREEN);  }  break;  case EVENT\_Motor\_Value :  currentDataPoint.valueMotor\_Value = dataPointPtr->valueMotor\_Value;  gizLog(LOG\_INFO,"Evt: EVENT\_MotorValue %d \n", currentDataPoint.valueMotor\_Value  );  if(0x01 == currentDataPoint.valueMotor\_Value)  {  //start motor  gpio\_high\_low(true,GPIO\_MOTOR);  }  else  {  //stop motor  gpio\_high\_low(false,GPIO\_MOTOR);  }  break;  ......    default:  break;  }  }  gizSendQueue(SIG\_UPGRADE\_DATA);    return 0;  }  gpio\_high\_low: mainly to control led or motor by setting the high/low level to the corresponding pin.  void gpio\_high\_low(bool on, uint8\_t pin\_gpio)  {    if (on)  {  qapi\_TLMM\_Drive\_Gpio(gpio\_id\_tbl[pin\_gpio], gpio\_map\_tbl[pin\_gpio].gpio\_id, QAPI\_GPIO\_LOW\_VALUE\_E);  }  else  {  qapi\_TLMM\_Drive\_Gpio(gpio\_id\_tbl[pin\_gpio], gpio\_map\_tbl[pin\_gpio].gpio\_id, QAPI\_GPImotor\_gpio\_configO\_HIGH\_VALUE\_E);  }  } | | |
| **Usage Instructions** | 1. Download code from github according to the repository in ”<https://github.com/ThunderSoft-XA/demo-Smart-vibrator>"sheet. 2. Compile the code and flash the image to Gokit4 development kit. 3. USB data cable to connect PC and Gokit development board. 4. Open the serial debugging assistant. 5. Click on "Device log" in Gizwits cloud to check if device is online. 6. When device online, you can click "view" button, and you'll see data from local was sent to Gizwits cloud. 7. Download app from "<https://download.gizwits.com/zh-cn/p/98/99>", and install it to your phone. 8. Open the apk, and you'll see "my device", here you can find your device by scanning QR code. 9. Next, generate the QR code. 10. open "gizwits debugging assistant\_v2.3.9/GIZ\_SerialTool.exe" 11. choose "small tools", click "QR code generation", input "product key" and "MAC/IMEI" 12. click "get QR code", and wait for a moment,you can see a QR code. 13. Click "my device" , and scan the generated QR code, you can find your device. 14. Then you can control the device and view the data. | | |
| **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.*