Qualcomm Developer Project  
Smart-Heart-rate-detector

|  |  |  |  |
| --- | --- | --- | --- |
| **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)*** | Using heart rate sensor to detect heart rate, and when the detected data is higher than the set threshold, the LED light is illuminated. | | |
| **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.** ”]  **Pluse\_Sensor\_374x467.png**  **PluseSensor_374x467**  [alt tag: “**using PluseSensor to detect heart rate value.** ”]  **DuPont\_ line\_578x305.png**  **DuPont_ line_578x305**  [alt tag: “**used as a cable**”]  **led\_348x225.png**  **led_348x225**  [alt tag: “**As a signal light, when the heart rate value exceeds the threshold, it is lit**”] | | |
| **Objective**   * *What inspired you to create this project?* * *What is your desired outcome?* | Mainly to connect the heart rate sensor to the ADC interface provided by the Gokit development board to collect the heart rate information. When the heart rate value is higher than the threshold, the LED light is illuminated. | | |
| **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   Robotics   Sensors   Smart Cities   Toys |
| **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> | |
| Pluse Sensor | <https://item.taobao.com/item.htm?spm=2013.1.0.0.781f5b59f8GSny&id=40354425711> | |
| Led | <https://detail.tmall.com/item.htm?spm=a230r.1.14.6.492b57c9IZIW9W&id=42895674142&cm_id=140105335569ed55e27b&abbucket=6> | |
| Dupoint line | <https://detail.tmall.com/item.htm?spm=a220m.1000858.1000725.6.3bb324e7xTOxcA&id=41065178536&skuId=83326522222&areaId=610100&user_id=2155882228&cat_id=2&is_b=1&rn=5dd30edec82ae5abc8d4645ec15c6aee> | |
| **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-Heart-rate-detector> | |
| **Additional Resources**  *List related links or resources such as websites, videos, presentations, or other materials* | Resource Title | Link or File Name (and provide file) | |
| [Video of "demo-Smart-Heart-rate-detector”](https://pan.baidu.com/s/1dDjGoV6fmwGDlkZc9CzZ0Q) | <https://pan.baidu.com/s/19pshLEQ899NDFtulEwMknQ>  6u5t | |
| **Build / Assembly Instructions** | Parts used Below are the items used in this project.  **parts.png**   |  |  | | --- | --- | | 2018-12-29 17:26:51屏幕截图 | 2018-12-29 17:24:00屏幕截图 |   [alt tag: “**Parts used for the Gokit4 development kit demo-Smart-Heart-rate-detector**”]   1. Win7 PC. 2. Pluse Sensor, used to collect heart rate information. 3. Led,When the detected heart rate value exceeds the set threshold, light the led. 4. Dupoint line, Used to connect other components as wires. 5. Oscilloscope,the oscilloscope is used to view the detected heart rate waveform in real time.   Deploying the project   1. According to the demo requirements, combine the onboard resources of the Gokit development board to purchase suitable devices. 2. Test if the optional device is available. 3. Build and debug hardware circuits. 4. Create a demo-Smart-Heart-rate-detector project locally. 5. Transplant relevant source code. 6. Develop related functions for heart rate calculations. 7. Joint debugging. 8. Code uploaded to github.   How does it work?  \After the above steps, user area data will be uploaded to the Gizwits cloud , below are some usage instructions to test the project.  Now let's introduce the demo-smart-heart-rate-detector’ s workflow.  gagentMain---->sensorInit----->led\_init---->Pulsesensor\_init.  demo-Smart-Heart-rate-detector/main/main.c  void gagentMain(void)  {  getFreeHeap();  sensorInit();  gizwitsInit();  timer\_init();  timer\_start();  }  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,  void sensorInit(void)  {  gizLog(LOG\_INFO, "Sensor initialization ...\n");  led\_init();  Pulsesensor\_init();  }  void led\_init()  {  gizLog(LOG\_INFO, "in led init...\n");  led\_gpio\_config();  led\_on\_off(false, led\_red);  }  void Pulsesensor\_init()  {  qapi\_Status\_t status = QAPI\_ERROR;  const char \*Channel\_Name\_ADC0 = ADC\_INPUT\_ADC0;  qapi\_Timer\_Sleep(2, QAPI\_TIMER\_UNIT\_SEC, true);  status = adc\_open\_handle();  if(status != QAPI\_OK)  {  //IOT\_DEBUG("Get ADC Handle ERROR!");  gizLog(LOG\_INFO,"adc open handle error...\n");  return;  }  status = adc\_get\_properties(Channel\_Name\_ADC0, &Properties\_ADC0);  if(status != QAPI\_OK)  {  //IOT\_DEBUG("Get ADC channel-%s Configuration ERROR!", Channel\_Name\_ADC1);  gizLog(LOG\_INFO,"Get ADC channel-%s Configuration ERROR...\n", Channel\_Name\_ADC0);  return;  }  }  demo-Smart-Heart-rate-detector/driver/timer/timer.c  qapi\_Status\_t timer\_init(void) //init  {  qapi\_Status\_t status = QAPI\_OK;  memset(&timer\_def\_attr, 0, sizeof(timer\_def\_attr));  timer\_def\_attr.cb\_type = QAPI\_TIMER\_FUNC1\_CB\_TYPE;  timer\_def\_attr.deferrable = false;  timer\_def\_attr.sigs\_func\_ptr = timer1\_handler;  timer\_def\_attr.sigs\_mask\_data = 0x11;  status = qapi\_Timer\_Def(&timer\_handle, &timer\_def\_attr);  return status;  }  qapi\_Status\_t timer\_start(void)  {  qapi\_Status\_t status = QAPI\_OK;  memset(&timer\_set\_attr, 0, sizeof(timer\_set\_attr));  timer\_set\_attr.reload = 100;  timer\_set\_attr.time = 10;  timer\_set\_attr.unit = QAPI\_TIMER\_UNIT\_MSEC;  status = qapi\_Timer\_Set(timer\_handle, &timer\_set\_attr);  return status;  }  void timer1\_handler(uint32\_t data) // timer callback  {  static bool led\_red\_status = true;  getHeartRateValue(&heartrate); //get heartvalue  if(!heartrate) //no value  {  return;  }  else if(((heartrate > HEART\_RATE\_THRESHOLD\_HIGH) || (heartrate < HEART\_RATE\_THRESHOLD\_LOW))) // value is not vaild  {  led\_on\_off(true, led\_red); //red on  //led\_on\_off(led\_red\_status, led\_red);  //led\_red\_status = !led\_red\_status;  }  else // value is vaild  {  led\_on\_off(false,led\_red); //red off  }  }  demo-Smart-Heart-rate-detector/driver/plusensor/plusensor.c  uint8\_t getHeartRateValue(uint32\_t\* heartrate ) //Algorithm implementation for calculating heart rate  {  gizLog(LOG\_INFO,"in getHeartRateValue...\n");  .......  memset(&result, 0, sizeof(result));  status = qapi\_ADC\_Read\_Channel(adc\_handle, &Properties\_ADC0, &result); // read the Pulse Sensor  ........  sampleCounter += 10; // keep track of the time in mS with this variable  Num = sampleCounter - lastBeatTime; // monitor the time since the last beat to avoid noise  // find the peak and trough of the pulse wave  if(Signal < thresh && Num > (IBI/5)\*3) // avoid dichrotic noise by waiting 3/5 of last IBI  {  if(Signal < T) // T is the trough  {  T = Signal; // keep track of lowest point in pulse wave  gizLog(LOG\_INFO,"Find trough, T = %d\n", T);  }  }  if(Signal > thresh && Signal > P) // thresh condition helps avoid noise  {  P = Signal; // P is the peak,keep track of highest point in pulse wave  gizLog(LOG\_INFO,"Find peak, P = %d\n", P);  }  // NOW IT'S TIME TO LOOK FOR THE HEART BEAT  // signal surges up in value every time there is a pulse  if (Num > 600)  { // avoid high frequency noise  if ( (Signal > thresh) && (Pulse == false) && (Num > (IBI/5)\*3) )  {  Pulse = true; // set the Pulse flag when we think there is a pulse  IBI = sampleCounter - lastBeatTime; // measure time between beats in mS  lastBeatTime = sampleCounter; // keep track of time for next pulse  if(secondBeat) // if this is the second beat, if secondBeat == TRUE  {  secondBeat = false; // clear secondBeat flag  for(int i=0; i<=9; i++)  {  rate[i] = IBI; // seed the running total to get a realisitic BPM at startup  }  }  if(firstBeat) // if it's the first time we found a beat, if firstBeat == TRUE  {  firstBeat = false; // clear firstBeat flag  secondBeat = true; // set the second beat flag  return 0; // IBI value is unreliable so discard it  }  // keep a running total of the last 10 IBI values  runningTotal = 0; // clear the runningTotal variable  for(int i=0; i<=8; i++)  {  rate[i] = rate[i+1]; // shift data in the rate array  // and drop the oldest IBI value  runningTotal += rate[i]; // add up the 9 oldest IBI values  }  rate[9] = IBI; // add the latest IBI to the rate array  runningTotal += rate[9]; // add the latest IBI to runningTotal  runningTotal /= 10; // average the last 10 IBI values  BPM = 60000/runningTotal; // how many beats can fit into a minute? that's BPM!  \*heartrate = BPM;  gizLog(LOG\_INFO,"BPM = %d\n", \*heartrate);  QS = true; // set Quantified Self flag  // QS FLAG IS NOT CLEARED INSIDE THIS ISR  }  }  if (Signal < thresh && Pulse == true)  { // when the values are going down, the beat is over  Pulse = false; // reset the Pulse flag so we can do it again  amp = P - T; // get amplitude of the pulse wave  thresh = amp/2 + T; // set thresh at 50% of the amplitude  P = thresh; // reset these for next time  T = thresh;  }  if (Num > 2500)  { // if 2.5 seconds go by without a beat  thresh = 512; // set thresh default  P = 512; // set P default  T = 512; // set T default  lastBeatTime = sampleCounter; // bring the lastBeatTime up to date  firstBeat = true; // set these to avoid noise  secondBeat = false; // when we get the heartbeat back  }  } | | |
| **Usage Instructions** | 1. Downloading code from github according to the repository in “<https://github.com/ThunderSoft-XA/demo-Smart-Heart-rate-detector>” sheet. 2. Compile the code and flash the image to Gokit4 development kit. 3. Connect the heart rate sensor to the ADC1 channel of the Gokit development board. 4. Connect one pin of the led to the D9 pin of the development board, and the other pin is connected to the vcc. 5. Turn on the oscilloscope, adjust to the appropriate gear position, connect the heart rate sensor's pulse signal output pin and GND to the oscilloscope. 6. USB data cable to connect PC and Gokit development board. 7. Touch your finger to the signal acquisition surface of the heart rate sensor. 8. Open the serial debugging assistant, you can see the collected data in real time. 9. When the detected data exceeds the threshold, you can see that the led is lit. | | |
| **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.*