|  |
| --- |
| **:: Chat program for blind people (can’t read or type but can listen)** |
| **Received message in voice <>Write message by voice** |
| **Programming language** | **Description**  |
| #define CUSTOM\_SETTINGS#define INCLUDE\_TEXT\_TO\_SPEECH\_SHIELD#define INCLUDE\_VOICE\_RECOGNIZER\_SHIELD | Define function that need to use in program |
| #include <OneSheeld.h>#include<SPI.h>#include"nRf24L01.h"#include"RF24.h"#include"printf.h" | Include library to be use in program |
| RF24 radio(7, 8); | Set CE and CSN pin for radio communication |
| const uint64\_t pipes[2] = { 0xF0F0F0F0E1LL, 0xF0F0F0F0D2LL }; | Set special network for radio communication  |
| bool go\_to\_tx\_mode = false;char input[32] = ""; | Define variable that need to be use in program |
| void setup() | Set command to run program |
| { OneSheeld.begin(); Serial1.begin(230400); printf\_begin(); radio.begin(); radio.setRetries(15, 15); radio.openWritingPipe(pipes[1]); radio.openReadingPipe(1, pipes[0]); radio.startListening(); radio.printDetails();} | * Begin the 1sheeld module operation
* Begin the Arduino serial monitor
* Start printing data
* Start radio function
* Number of retry before error
* radio channel for transmitting
* radio channel for receiving
* radio begin as receiver
* print radio details
 |
| void loop() | Put program to continuously run |
| { if(VoiceRecognition.isNewCommandReceived()) {char\* input=VoiceRecognition.getLastCommand();  go\_to\_tx\_mode = true; if (go\_to\_tx\_mode) { radio.stopListening(); bool ok = radio.write(input, 32);  if (ok) { printf("ME : %s\n", input); go\_to\_tx\_mode = false; radio.startListening(); } else printf("could not write....\n"); } } | * Check if there is new voice message

**If voice message present:*** Get voice message and save as character array
* Turn radio into transmitter mode

**If radio is in transmitter mode:*** Stop receiving data from the other radio
* Send voice message to the other radio

**If message sent*** Print sent message (for reference)
* Turn back radio to receiver mode
* Start receiving data

**If message not sent*** Print error message
 |
| else if (radio.available()) { char payload[32] = ""; bool done = false; while (!done) { done = radio.read( payload , 32 ); printf("HIM : %s\n", payload); TextToSpeech.say(payload); } }} | **If no voice message and radio is available*** Define input variable input (max bit=32)
* Define command

**While input message does not exceed 32 bits*** Read input from the other radio
* Print input message from the other radio
* Convert text into speech
 |

|  |
| --- |
| **:: Chat program for deaf people (Can read and type but cannot listen)** |
| **Received message in text <>Write message by text** |
| **Programming language** | **Description**  |
| #include <OneSheeld.h>#include<SPI.h>#include"nRf24L01.h"#include"RF24.h"#include"printf.h" | Include library to be use in program |
| RF24 radio(7, 8); | Set CE and CSN pin for radio communication |
| const uint64\_t pipes[2] = { 0xF0F0F0F0E1LL, 0xF0F0F0F0D2LL }; | Set special network for radio communication  |
| bool go\_to\_tx\_mode = false;char input[32] = ""; | Define variable that need to be use in program |
| void setup() | Set command to run program |
| {  Serial1.begin(230400); printf\_begin(); radio.begin(); radio.setRetries(15, 15); radio.openWritingPipe(pipes[1]); radio.openReadingPipe(1, pipes[0]); radio.startListening(); radio.printDetails();} | * Begin the Arduino serial monitor
* Start printing data
* Start radio function
* Number of retry before error
* radio channel for transmitting
* radio channel for receiving
* radio begin as receiver
* print radio details
 |
| void loop() | Put program to continuously run |
| if (Serial.available()) { int len = Serial.readBytesUntil('\n', input, 31); input[len] = '\0'; go\_to\_tx\_mode = true; if (go\_to\_tx\_mode) { radio.stopListening(); bool ok = radio.write(input, 32);  if (ok) { printf("ME : %s\n", input); go\_to\_tx\_mode = false; radio.startListening(); } else printf("could not write....\n"); } } | **Check if there is new text messages from serial monitor, if there is new text:*** read the text until 32 bits
* set length to the text
* Turn radio into transmitter mode

**If radio is in transmitter mode:*** Stop receiving data from the other radio
* Send text message to the other radio

**If message sent*** Print sent text message
* Turn back radio to receiver mode
* Start receiving data

**If message not sent*** Print error message
 |
| else if (radio.available()) { char payload[32] = ""; bool done = false; while (!done) { done = radio.read( payload , 32 ); printf("HIM : %s\n", payload); } }} | **If no text message and radio is available*** Define input variable input (max bit=32)
* Define command

**While input message does not exceed 32 bits*** Read input from the other radio
* Print input message from the other radio
 |