

৯/ দুটি JRC বোর্ডের মধ্যে ব্লুটুথ কমিউনিকেশন:

প্রয়োজনীয় উপকরণ:

১/ এলইডি লাইট

২/ ব্রেডবোর্ড

৩/ জাম্পার ওয়্যার

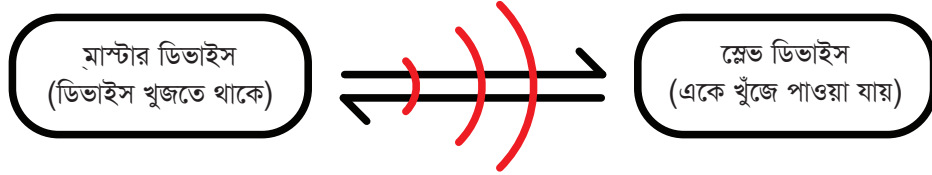
৪/ JRC বোর্ড ২ টি

৫/ ২২০ওহম রেজিস্টর ও ১০কিলো ওহম রেজিস্টর

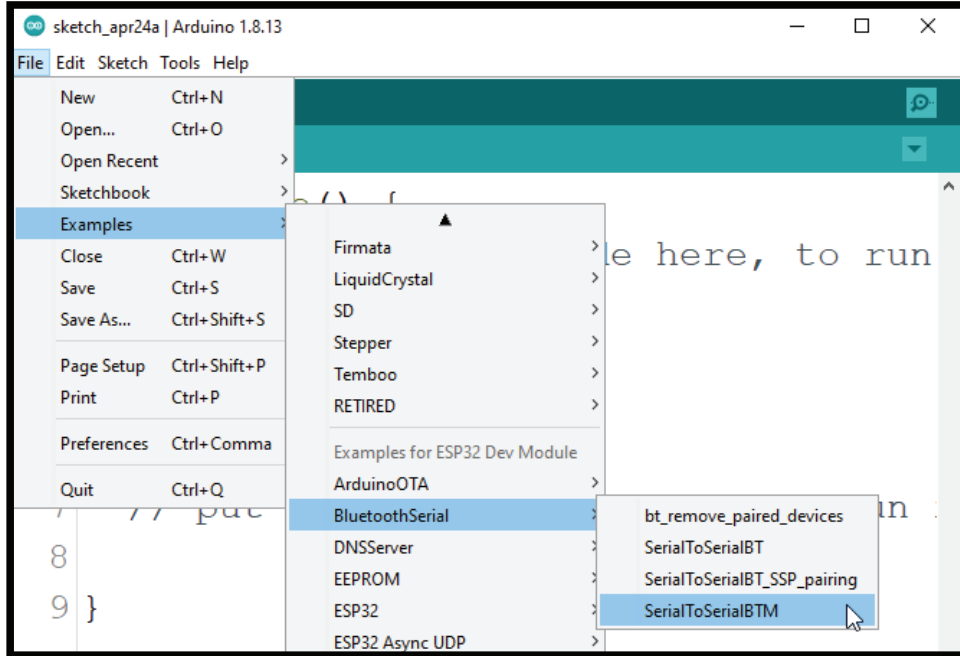
৬/ পুশ বাটন সুইচ

বর্ণনাঃ পূর্বের প্রজেক্টে আমরা JRC বোর্ড এবং মোবাইল ফোনের মধ্যে যোগাযোগ স্থাপন করে দেখিয়েছি কিভাবে একটি স্মার্টফোন এর মাধ্যমে বিভিন্ন ডিভাইস নিয়ন্ত্রণ করা যায়। এবার আমরা দেখবো দুটি JRC বোর্ডের মধ্যে কিভাবে যোগাযোগ করতে হয়। এবং এখানে চাইলে এটি বোর্ড কে রিমোট হিসেবে ব্যবহার করে আরেকটি বোর্ড কে কোন কাজে নিয়ন্ত্রণ করা সম্ভব।

প্রথমত ব্লুটুথ কমিউনিকেশন করতে গেলে অবশ্যই একটি ডিভাইস কে মাস্টার মুডে এবং আরেকটি ডিভাইসকে স্লেভ (রিসিভার) মুডে রাখতে হয়। এখানে যে ডিভাইসটি মাস্টার মুডে থাকে, তার কাজ হলো নির্দিষ্ট ব্লুটুথ ডিভাইসকে খুঁজে যাওয়া, এবং স্লেভ ব্লুটুথ ডিভাইসটি রিসিভার মুডে থাকে এবং তাকে কানেক্ট করতে এলাউ করে থাকে। যেমন আমরা পূর্বে যে প্রজেক্ট করেছিলাম, সেখানে আমাদের স্মার্টফোন ছিলো মাস্টার মুডে এবং JRC বোর্ডটি ছিলো রিসিভার মুডে। যে কারণে আমরা ফোন থেকে সার্চ করে ব্লুটুথ ডিভাইসকে খুঁজে পেয়েছি এবং কানেক্ট করতে পেরেছি। যদি JRC বোর্ড কে মাস্টার মুডে নেয়া হয়, তবে তাকে অন্য কোন ডিভাইস থেকে সার্চ করা হলে খুঁজে পাওয়া যাবেনা।



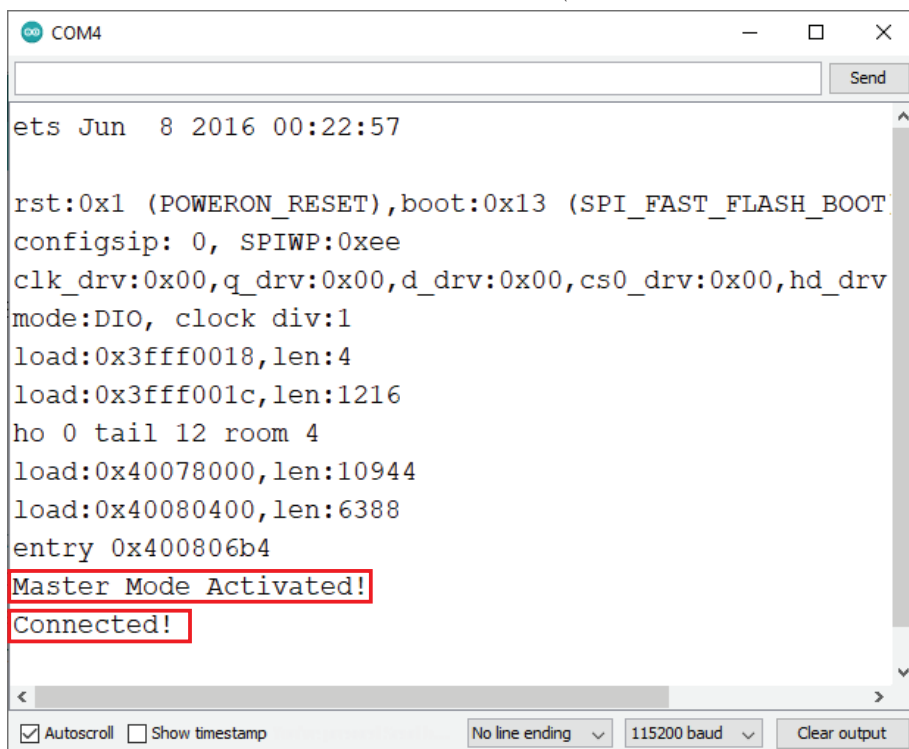
এখানে ব্যাপারটি এমন নয় যে মাস্টার ডিভাইস শুধু ডাটা পাঠিয়েই যাবে এবং স্লেভ ডিভাইস ডাটা রিসিভ করে যাবে। এখানে যেকোন ডিভাইস ডাটা পাঠাতে পারবে এবং রিসিভও করতে পারবে। এখন কথা হলো JRC বোর্ডে কিভাবে আমরা মাস্টার মুড এন্টিভেট করবো। এর জন্যেও এক্সাম্পল কোড রয়েছে যা তোমরা file -> examples -> BluetoothSerial -> SerialToSerialBTM এই অপশনে পাবে।



এই অপশনে একটি কোডের ইউন্ডো ওপেন হবে যা দেখতে পূর্বে আমরা যা দেখে এসেছি প্রায় সেরকমই। এখানে কোডের মূল দরকারী অংশটুকু নিচে তুলে ধরা হচ্ছেঃ

```
#include "BluetoothSerial.h"
BluetoothSerial SerialBT;
String name = "ESP32test";
bool connect;
void setup(){
    Serial.begin(115200);
    SerialBT.begin("ESP32test", true);
    Serial.println("Master Mode Activated!");
    connect = SerialBT.connect(name);
    if(connect) Serial.println("Connected!");
}
void loop(){
    if(Serial.available()) {
        SerialBT.write(Serial.read());
    }
    if(SerialBT.available()) {
        Serial.write(SerialBT.read());
    }
    delay(20);
}
```

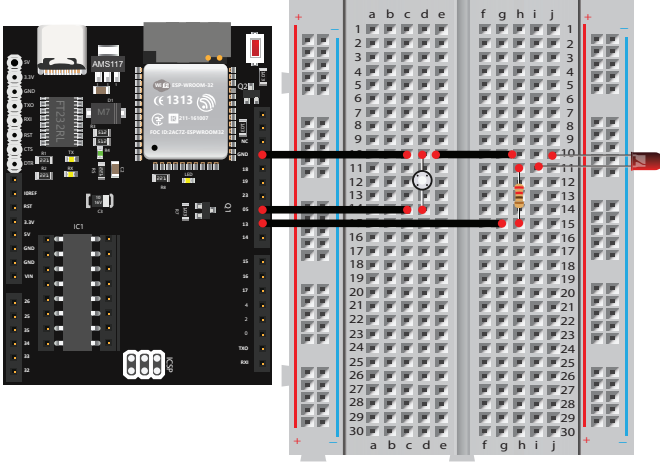
তোমরা লক্ষ্য করলে দেখতে পাবে আমরা পূর্বে JRC বোর্ডকে স্লেভ ডিভাইস হিসেবে ব্যবহার করার সময় যে কোড দিয়েছিলাম, তার সাথে এটি অনেকটাই মিলে যায়। কেবল কিছু জায়গায় ছোট পার্থক্য আছে। প্রথমত, আমরা আগে স্লেভ ডিভাইসে `SerialBT.begin("ESP32test");` লিখে ব্লুটুথ ইনিশিয়েট করতাম, সেই জায়গায় এখন `SerialBT.begin("ESP32test", true);` লেখার ফলে এটি মাস্টার মুডে অ্যাক্টিভেট হয়ে যাবে। বাদবাকি ফাংশন ঠিকই আছে, এখানে কেবল নতুন একটি ফাংশন এর ব্যবহার হয়েছে যেটি হল `SerialBT.connect()`। এটির কাজ হলো অন্য একটি স্লেভ ডিভাইস এর সাথে সংযুক্ত হবার চেষ্টা করা। এখন কথা হলো কোন স্লেভ ডিভাইস এর সাথে যুক্ত হতে হবে সেটা সে কিভাবে বুঝতে পারবে? এর জন্য এই ফাংশনের ভেতরে প্রথমে যে স্লেভ ডিভাইস এর সাথে যুক্ত করতে চাই সেই ডিভাইসের নাম টি লিখে দেয়া লাগে। যেমন আমরা যদি পূর্বের স্লেভ ডিভাইসটিই ব্যবহার করতে চাই যেখানে এর নাম ছিলো "ESP32test", সেক্ষেত্রে সেই নামটিই এখানে ইনপুট দিতে হবে। ঝামেলা এড়াতে আমরা শুরুতে "name" নামে একটি স্ট্রিং ভ্যারিয়েবল ডিক্লেয়ার করে নিয়েছি যার ভেতরে এই নামটি সেভ করে রাখা হয়েছে। যারা জানোনা তাদের উদ্দেশ্যে, স্ট্রিং হলো একপ্রকার বিশেষ ক্যারেক্টার ভ্যারিয়েবল অ্যারে, যেখানে একসাথে একের অধিক ক্যারেক্টার সেভ করে রাখা যায় এবং সেটি সরাসরি ব্যবহার করা যায়। সুতরাং আমরা `SerialBT.connect(name)` লেখার মাধ্যমে ঐ নির্দিষ্ট নামে থাকা স্লেভ ডিভাইসটিকে সার্চ করতে থাকে এবং কানেক্ট করার চেষ্টা করে। যদি এই কানেকশন সফল হয়, তবে এটি 1 ভ্যালু রিটার্ন করে যা আমরা আরেকটি ভ্যারিয়েবলে সেভ করে রাখছি যার নাম হচ্ছে "connect"। এখানে "connect" নামক ভ্যারিয়েবল এর টাইপ হিসেবে bool ব্যবহার করা হয়েছে যার পূর্ণ মানে হচ্ছে বুলিয়ান। এটিতে কেবল এক বিট মেমোরি সেভ করা সম্ভব অর্থাৎ এতে হয় ০ নয়তো ১ সেভ করে রাখা যায়। যেটি আমরা একটি কন্ডিশন স্টেটমেন্ট এ ব্যবহার করে সেখানে চেক করতে পারি যে কানেকশন আসলেই সফল হয়েছে কিনা। উল্লেখ্য, if() স্টেটমেন্ট এর ভেতরে 1 লেখা মানে এর শর্তকে সত্য হিসেবে ধরে নেয়া। ১ মানে সত্য, ০ মানে মিথ্যা। বাদবাকি সবকিছুই আগের মতোই থেকে যাবে। এবার আমরা এই কোড টি পরীক্ষা করে দেখার জন্য দুটি JRC বোর্ড নিয়ে একটি তে এই মাস্টার এর কোড এবং অপরটিতে স্লেভ ডিভাইস এর কোড আপলোড করি। উল্লেখ্য যে দুটি JRC বোর্ডে একই সাথে সিরিয়াল মনিটর ওপেন করে সম্ভব না। এক্ষেত্রে দুটি আলাদা কম্পিউটার লাগবে অথবা আমরা শুধু মাস্টার ডিভাইসের সাথে কম্পিউটার সংযোগ রেখে সিরিয়াল মনিটর ওপেন করতে পারি আর অপরটিতে সাধারণ পাওয়ার ব্যাংক থেকে ইউএসবি ক্যাবল কিংবা ব্যাটারির মাধ্যমে চালু রেখে এটা পরীক্ষা করে দেখতে পারি যে মাস্টার ডিভাইস থেকে সংযোগ সফল হইয়েছে কিনা। সংযোগ সফল হলে সিরিয়াল মনিটরে নিচের মতো আউটপুট দেখাবে:



```
ets Jun  8 2016 00:22:57

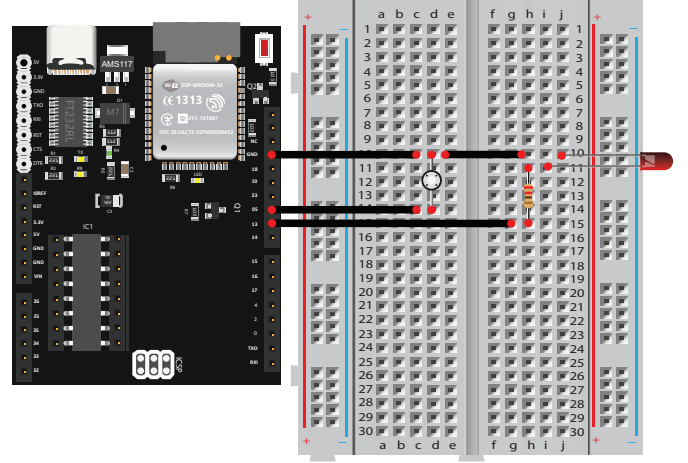
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1216
ho 0 tail 12 room 4
load:0x40078000,len:10944
load:0x40080400,len:6388
entry 0x400806b4
Master Mode Activated!
Connected!
```

আমরা জানতে পারলাম যে কিভাবে দুটি JRC বোর্ডের মধ্যে সংযোগ স্থাপন করা যায়। একবার সংযোগ স্থাপন হয়ে গেলে এরপর দুই প্রান্ত থেকেই আগের মতো কথপোকথন চালানো সম্ভব। এবার তাইলে এক মজার কাজ করা যাক। আমরা দুটি বোর্ডেই একটি করে বাটন এবং একটি করে লাইট স্থাপন করি। এক বোর্ডে বাটন প্রেস করলে আরেক বোর্ডের লাইট জ্বলে উঠবে এবং অপর বোর্ডের বেলাতেও সেম লজিক খাটবে। এর মানে আমরা দুটি বোর্ডের মধ্যে একটা যোগাযোগ ব্যবস্থা দাড় করাতে যাচ্ছি। এক্ষেত্রে নিচের সার্কিটটি তৈরী করতে পারো:



MASTER DEVICE

```
#include "BluetoothSerial.h"
BluetoothSerial SerialBT;
bool connect, state;
void setup(){
  pinMode(5, INPUT_PULLUP);
  pinMode(13, OUTPUT);
  Serial.begin(115200);
  SerialBT.begin("ESP32test", true);
  Serial.println("Master Mode Activated!");
  connect = SerialBT.connect("ESP32test");
  if(connect) Serial.println("Connected!");
}
void loop(){
  if(SerialBT.available()) {
    char x = SerialBT.read();
    if(x=='A') digitalWrite(13,HIGH);
    else if(x=='B') digitalWrite(13,LOW);
  }
  else if(digitalRead(5) == 1 && state != 1){
    state = 1; SerialBT.write('B');
  }
  else if(digitalRead(5) == 0 && state != 0){
    state = 0; SerialBT.write('A');
  }
}
```



SLAVE DEVICE

```
#include "BluetoothSerial.h"
BluetoothSerial SerialBT;
bool state;
void setup(){
  pinMode(5, INPUT_PULLUP);
  pinMode(13, OUTPUT);
  Serial.begin(115200);
  SerialBT.begin("ESP32test");
  Serial.println("Slave Mode Activated!");
}
void loop(){
  if(SerialBT.available()) {
    char x = SerialBT.read();
    if(x=='A') digitalWrite(13,HIGH);
    else if(x=='B') digitalWrite(13,LOW);
  }
  else if(digitalRead(5) == 1 && state != 1){
    state = 1; SerialBT.write('B');
  }
  else if(digitalRead(5) == 0 && state != 0){
    state = 0; SerialBT.write('A');
  }
}
```

এখানে তোমরা দুটি বোর্ডে হুবহু একই সার্কিট ব্যবহার করছো যে সার্কিট তুমি আগেই পুশ বাটন দিয়ে এলইডি লাইট জ্বালানোর প্রজেক্টে ব্যবহার করেছিলে। এখানে কেবল পার্থক্য হলো কোন বোর্ডের নিজের সুইচে নিজের লাইট টি জ্বলবেনা, অপর বোর্ডের লাইট জ্বালাবে। যেহেতু ব্লুটুথ এর মাধ্যমে কমিউনিকেশনের কাজ করা হচ্ছে তাই এখানে একটি বোর্ড কে অবশ্যই মাস্টার মুডে এবং অপরটিকে স্লেভ মুডে ব্যবহার করতে হবে। এর জন্য কোডের সেটাপ অংশে একটু পার্থক্য রয়েছে যেগুলো সম্পর্কে তুমি আগেই শিখে এসেছো। পুশ বাটন থেকে কিভাবে রিডিং নিতে হয় এবং লাইট কিভাবে জ্বালাতে হয় এগুলোও আগে শিখে এসেছো, তাইনা? এখানে কিছুটা নতুন সিস্টেম প্রণয়ন করা হয়েছে যা কিনা ব্লুটুথ কমিউনিকেশনের বেলায় জরুরি। ব্লুটুথ কমিউনিকেশন এ একটানা খুব দ্রুত ডাটা পাঠাতে থাকলে রিসিভার প্রান্তে সে ডাটাগুলো জ্যাম হয়ে যায় এবং পরবর্তীতে কমিউনিকেশন সিস্টেম হ্যাং হয়ে যেতে পারে। তাই প্রয়োজন ব্যতিত অনবরত ডাটা পাঠানো যাবেনা। এখন আমরা এমন সিস্টেম করেছি যে যদি কোন বোর্ড এর বাটন প্রেস করা হয়, তবে সে 'A' ক্যারেঙ্টার পাঠাবে যেটা অপর ডিভাইসে রিসিভ করলে লাইট জ্বালিয়ে দিবে। আবার বাটন প্রেস ছেড়ে দিলে ঐ বোর্ডটি 'B' ক্যারেঙ্টার পাঠাবে যা কিনা অপর ডিভাইসে রিসিভ করলে লাইট বন্ধ করে দিবে। একই সিস্টেম অপর ডিভাইস থেকেও নিয়ন্ত্রণ করা সম্ভব। এখন এখানে বার বার বাটনপ্রেস এর কন্ডিশন চেক করে অনবরত ডাটা পাঠাতে থাকে, তাহলে তো সমস্যা। এক্ষেত্রে আমরা এমন সিস্টেম করতে পারি যে বাটন প্রেস করলে কেবল একবারই ডাটা পাঠাবে। এরপর চেপে ধরে রাখলে তো বাটন প্রেস এর রিডিং পরিবর্তন হচ্ছেনা। সেক্ষেত্রে নতুন করে কোন ডাটা পাঠাবেনা। যেইমাত্র বাটন প্রেস ছেড়ে দেয়া হবে, সাথে সাথে কন্ডিশন চেঞ্জ হবার কারণে নতুন করে 'B' ডাটা পাঠাবে এবং এরপর আবার প্রেস না করা পর্যন্ত কোন ডাটা পাঠাবেনা। এই কাজটির জন্য আমরা আলাদা করে একটি ভ্যারিয়েবল state নিয়েছি যার প্রয়োগ কোডে দেখানো হয়েছে। এর ফলে তোমার ডিভাইস সর্বক্ষণ লিসেনিং মুডে থাকবে এবং কমান্ড না দিলে কোন ডাটা পাঠাবেনা। এর ফলে খুব সুন্দর ডাটা কমিউনিকেশন সম্ভব এবং প্রজেক্ট টি সম্পূর্ণ হয়ে যায়।