SECURING CONNECTION BETWEEN ARDUINO USING BT

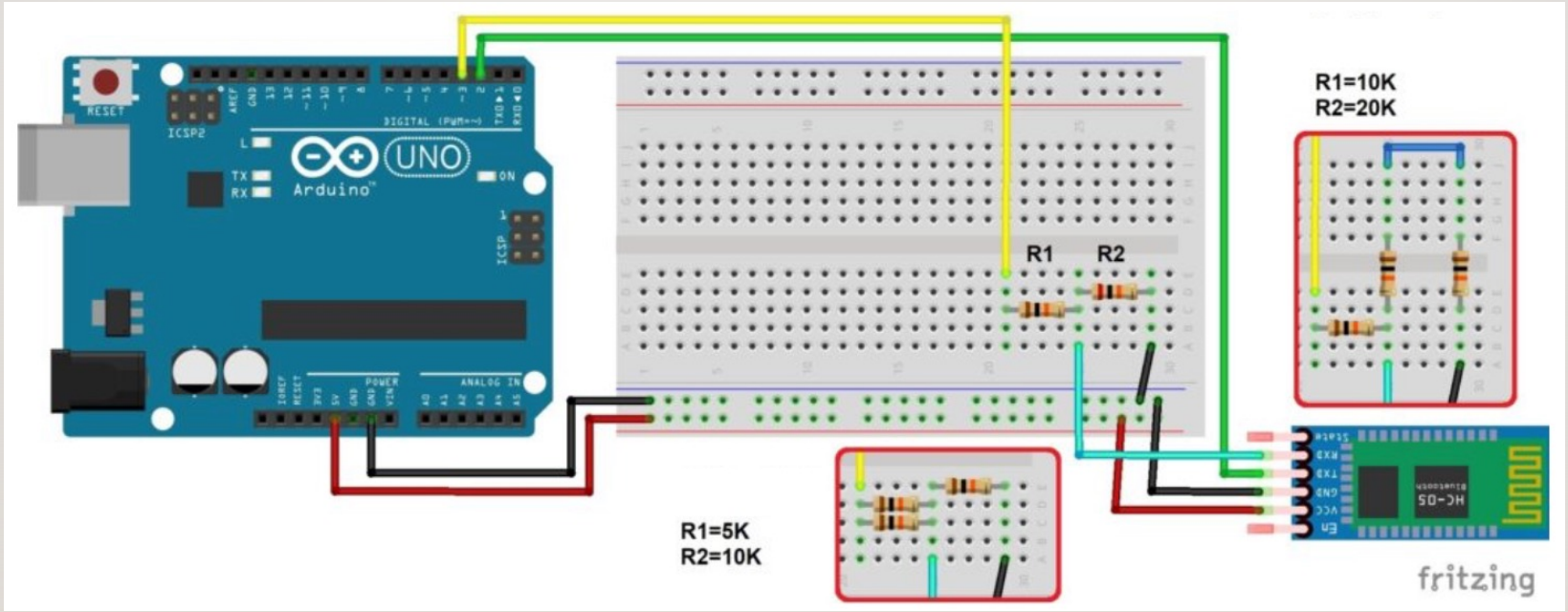
DR. BIAGIO BOI

IOT SECURITY LAB

CHECKLIST

- How to configure HT05 and arduino
- AT commands
- Bluetooth connection between two arduino
- Secure BT connection using AES

HOW TO CONFIGURE HT05 AND ARDUINO



HOW TO CONFIGURE HT05 AND ARDUINO

- Once done the wiring as in the previous slide, we need to specify to HT05 which mode use (AT COMMANDS or NORMAL).
- Since we need to configure the HT05 we have to give 5v to the EN (or KEY) of HT05.
- When we want to exit from AT COMMANDS mode, we need to connect the EN pin to the GND and restart the HT05.
- SoftwareSerial library will be used to implement UART connection.

AT COMMANDS

- AT commands are used to configure HT05, we will use only a subset of them for the basic configuration.
- Full commands: https://s3-sa-east-1.amazonaws.com/robocore-lojavirtual/709/HC-05_ATCommandSet.pdf
- We need to set the name, set the role, and in next slides set the binding address.
- AT (to check if the connection is ok)
- AT+NAME=SLAVE
- AT+ROLE=0 (for slave, 1 for master)

BLUETOOTH CONNECTION BETWEEN TWO ARDUINO

- Once configured the first arduino with the related HT05 module, we need to do the same operation on the other Arduino.
- This time, instead of defining it as slave, we have to define it as master and bind to the address of the other HT05 module.
- AT+NAME=MASTER
- AT+ROLE=1
- AT+BIND=<address of other HT05 module>
- Note: to get the address of one HT05 module, it's possible to send AT+ADDR?

SECURE BT USING AES

- We defined the connection, now we need to secure this connection using AES.
- We know that this is not the best security mechanism, but it's really useful in this context because it requires low effort to be executed.
- Download the AES library at https://forum.arduino.cc/t/new-aes-library/86966

```
byte set_key (byte key[], int keylen) ;
  void clean () ;  // delete key schedule after use
  void copy_n_bytes (byte * dest, byte * src, byte n) ;

  byte encrypt (byte plain [N_BLOCK], byte cipher [N_BLOCK]) ;
  byte cbc_encrypt (byte * plain, byte * cipher, int n_block, byte iv [N_BLOCK]) ;

  byte decrypt (byte cipher [N_BLOCK], byte plain [N_BLOCK]) ;
  byte cbc_decrypt (byte * cipher, byte * plain, int n_block, byte iv [N_BLOCK]) ;
```