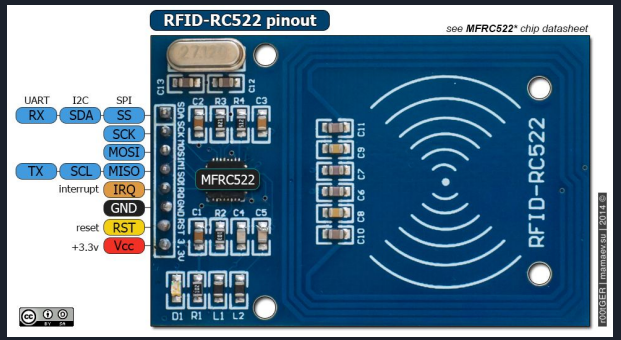
RFID Workshop

Led by Will McGloughlin, Cole Baughman, Amal Alsubaie, and Jackie Cheng

What it is

Radio-frequency identification (RFID) is a technology where we use radio waves to automatically identify or track objects.

- RFID Reader (we will be using **RFID–RC522**)
  - It generates radio waves continuously and waits to receive data from RFID tag
  - It uses microcontroller to process the data received.
- **RFID tag**
  - The tag has the data we want to read. It transmit the data when in range of the RFID Reader.
    - Active RFID tag: has its own power source
    - Passive RFID tag: gets power from the RFID reader; the reader radio waves induces current into the tag.
- Range and Frequency of operation:
  - Low Frequency range is up to 10cm
  - High Frequency range is up to 1m,
  - Ultra High Frequency range is from 10 to 15m



https://www.abr.com/what-is-rfid-how-does-rfid-work/
https://internetofthingsagenda.techtarget.com/definition/RFID-radio-frequency-identification
https://www.youtube.com/watch?v=Ukfpq71BoMo&ab_channel=ALLABOUTELECTRONICS

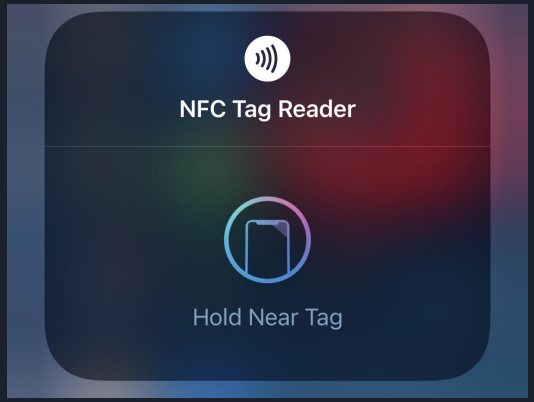
https://microcontrollerslab.com/rc522-rfid-reader-pinout-arduino-interfacing-examples-features/

How it works

The RFID reader sends radio waves non−stop with a specific frequency. These waves provide power to the RFID tag, synchronize the clocks, and carrys the data of the RFID tag back into the reader.

- For Low and High Frequency tags: Inductive Coupling or Near Field Coupling (NFC)
- For Ultra High Frequency tags: Electromagnetic Coupling or Far Field Coupling (FFC)

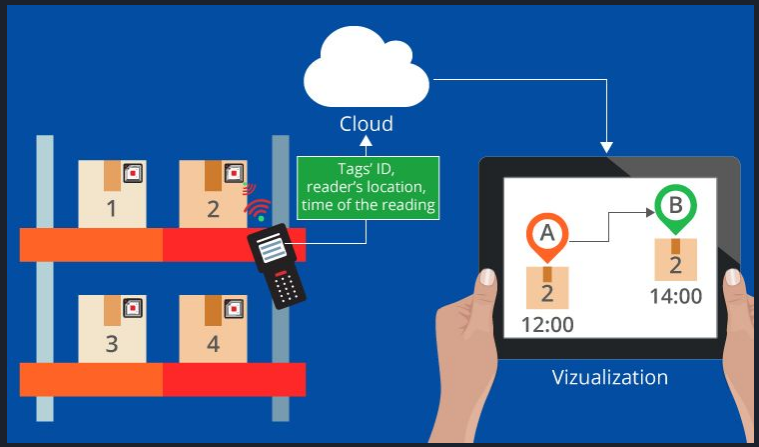
NFC is the technology used in most of our phones today!

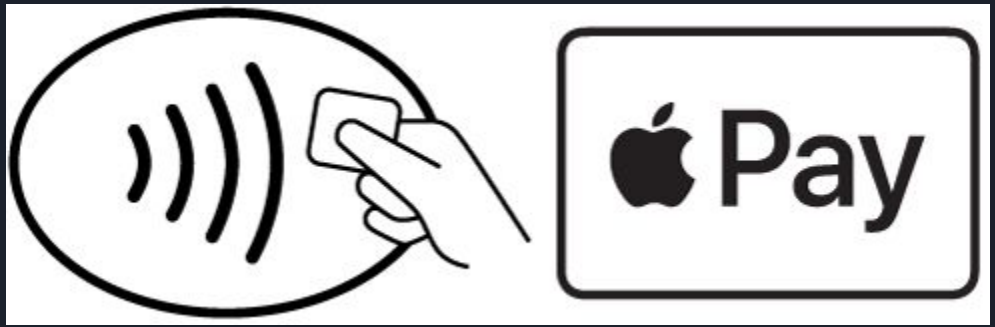


https://www.youtube.com/watch?v=Ukfpq71BoMo&ab_channel=ALLABOUTELECTRONICS

How its used

- Self-Checkout
- Touchless Payment
- Inventory Control
- Equipment Tracking
- Microchip Implant
- Toll Booth Payment





Programing the Arduino

- Arduino IDE supports a dialect of C++, **Arduino C**
- C is a lower level programing language compared to the languages you may know like Python or Java
  - Python allows us to be more flexible, for example variable types can be determined at runtime by the data assigned to them
  - C code lacks this flexibility
- C must be compiled and uploaded by the IDE to be run on Arduino
- We are going to go over some syntax you'll need to know for your development

C Syntax - Semicolons and Variable Types

- For this workshop we will be using statements similar in nature to ones you know from other programing languages
- Syntax of those statements will be different
- For example:
  - Every non blocking statement must end with a semicolon ;
    - This would not include statements like if or while, the are followed by braces {} containing the statements to be executed by them
  - Variables must be declared with a type
    - int x;
    - int y = 4;
    - float i = 3.4;

```
int spectrumValue[7];
int filterValue = 140;
int ledPinR = 10;
int ledPinG = 9;
int ledPinB = 11;
```

C Syntax - Calling Functions

- Calling functions is similar to Java
- We will need to supply them with arguments if the function requires them
- For example a function with two arguments
  - function_name(argument1, argument2);
- Calling a function with no arguments leaves the parentheses empty
  - function_name();

```
int i = 2;
int j = 3;
int k;

k = myMultiplyFunction(i, j); // k now contains 6
Serial.println(k);
delay(500);
```

Setup and Loop

- When creating a program the Arduino IDE automatically creates two functions
- setup() is a place for you to initialize your variables. **Runs once.**
- loop() continuously runs on your Arduino. **Runs after setup, forever.**

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

C Syntax - If-else Statements

- For this workshop need to know if statements
  - Similar syntax to Java
- Possible conditional operators used in this workshop include
  - `==` equals
  - `!=` not equal
  - `&&` and
  - `||` or
  - `>` greater than
  - `<` less than

```
if (temperature >= 70) {
  // Danger! Shut down the system.
}
else if (temperature >= 60) { // 60 <= temperature < 70
  // Warning! User attention required.
}
else { // temperature < 60
  // Safe! Continue usual tasks.
}
```

SPI

What is SPI?

Serial Peripheral Interface is a method of short-distance communication, often used with microcontrollers.

How does it work?

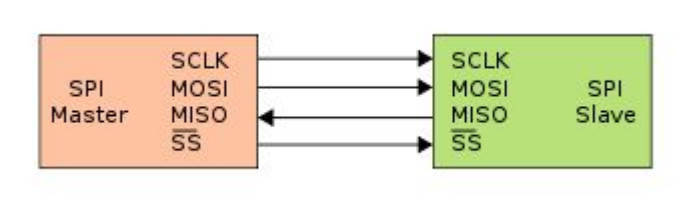
SCLK: Serial Clock (output from master)

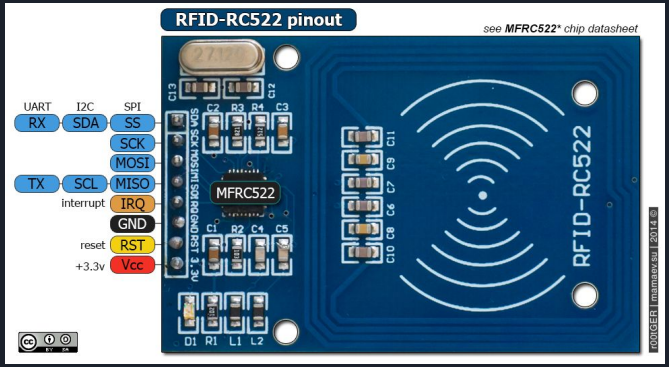
MOSI: Master Out Slave In (data output from master)

MISO: Master In Slave Out (data output from slave)

CS /SS: Chip/Slave Select (often active low, output from master to indicate that data is being sent)

It's important to note that only specific arduino pins can perform a given SPI logic signal. Planning ahead on circuit wiring is important!





Building the circuit

Signal	MFRC522 Reader/PCD Pin	Arduino Uno/101 Pin	Arduino Mega Pin	Arduino Nano v3 Pin
RST/Reset	RST	9	5	D9
SPI SS	SDA(SS)	10	53	D10
SPI MOSI	MOSI	11 / ICSP-4	51	D11
SPI MISO	MISO	12 / ICSP-1	50	D12
SPI SCK	SCK	13 / ICSP-3	52	D13