How Does an IR Remote Controller Work?

An **IR remote controller** works by transmitting encoded signals as **infrared light** to control a receiving device (e.g., TV, air conditioner). Here's how it operates:

1. **Button Press**:
    - When you press a button, the remote identifies the command associated with that button (e.g., volume up, power on).
2. **Data Encoding**:
    - The remote uses an encoding protocol (e.g., **NEC**, **Sony**, **RC5**) to convert the command into a series of binary data (0s and 1s).
3. **Modulation**:
    - The binary data is superimposed onto a carrier signal, typically at a frequency of **38 kHz**. This modulation helps the IR receiver differentiate the remote's signal from ambient light.
4. **Transmission**:
    - The remote's IR LED emits bursts of infrared light corresponding to the modulated binary data.
    - These bursts are invisible to the human eye but can be detected by an IR receiver.

How Is the Data Encoded?

IR remotes use specific encoding protocols to format the data. Each protocol has its way of organizing the bits and timing patterns. Here's a general overview:

1. **Start Bit**:
    - A unique signal indicates the start of a transmission, helping the receiver know when to begin decoding.
2. **Address Bits**:
    - These bits identify the device the command is intended for (e.g., TV, DVD player).
3. **Command Bits**:
    - These bits specify the action to be performed (e.g., volume up, channel down).
4. **Stop Bit**:
    - A signal to indicate the end of the transmission.
5. **Pulse Width Modulation**:
    - The binary data is encoded into patterns of ON/OFF pulses of the infrared light. For example:
        - A "1" might be a 38 kHz pulse for **1.2 ms** followed by no pulse for **0.6 ms**.
        - A "0" might be a 38 kHz pulse for **0.6 ms** followed by no pulse for **1.2 ms**.

Each protocol defines the exact timing and format for these pulses.

Is the Data for Every Button Saved in the Controller?

Yes, the data for every button is pre-programmed into the remote controller's memory:

1. **Button Mapping**:
    - Each button is mapped to a specific binary code representing its function (e.g., `0xFFA857` for "Volume Up").
    - These codes are stored in the controller’s firmware (non-volatile memory).
2. **Microcontroller**:
    - The remote contains a small microcontroller that looks up the stored code when a button is pressed.
    - The microcontroller retrieves the binary code for the button and sends it to the IR LED for transmission.
3. **Pre-Programming**:
    - Universal remotes may store codes for multiple brands and devices. Users select or program the device brand, and the remote sends commands compatible with that device.

Example of Data Encoding (NEC Protocol):

1. **Structure**:
    - **Start Pulse**: 9 ms ON, 4.5 ms OFF.
    - **Address**: 16 bits for the device address.
    - **Command**: 8 bits for the action.
    - **Inverted Command**: 8 bits to verify the command.
    - **Stop Pulse**: A short pulse to signal the end.
2. **Example Command**:
    - Button "Power On":
        - Address: `0x20DF` (binary: `00100000 11011111`).
        - Command: `0x10` (binary: `00010000`).
        - Inverted Command: `0xEF` (binary: `11101111`).
    - Transmitted Signal:
        - `9 ms ON`, `4.5 ms OFF`, followed by 32 bits of data.

Key Points:

- **Pre-Programmed Data**:
  - Each button has a specific code stored in the remote's memory.
  - Universal remotes can switch between code sets for different devices.
- **Encoding**:
  - Data is encoded into binary using protocols like NEC or Sony.
  - The binary data is modulated onto a 38 kHz carrier signal for reliable transmission.
- **Transmission**:
  - The IR LED emits pulses of infrared light that correspond to the binary code.
- **Decoding at the Receiver**:
  - The receiver demodulates the signal, interprets the binary data, and executes the command.

This architecture ensures compatibility and efficient communication between remotes and devices.

Test if your remote control sends an infrared (IR) signal

EEVblog #506 - IR Remote Control Arduino Protocol Tutorial

Flipper Zero: Beginner Guide