Sure! Let me explain what this Arduino code does, step by step:

**🔌 Libraries and Setup**

#include <Wire.h>

* Includes the **Wire** library, which allows the Arduino to communicate using **I2C** (a two-wire communication protocol).

void setup() {

 Serial.begin(115200);

 Wire.begin();

}

* Serial.begin(115200);: Initializes serial communication with the computer at 115200 bits per second.
* Wire.begin();: Initializes the Arduino as an **I2C master**, ready to send data over the I2C bus.

**🔁 Main Loop**

void loop() {

 while (Serial.available() > 0) {

 int deviceAddr = Serial.parseInt();

 int blinkCnt = Serial.parseInt();

* Serial.available() > 0: Checks if there is any data coming in from the serial port.
* Serial.parseInt(): Reads the next integer from the serial input. It’s used **twice** here to:
	1. Get the **I2C address** of the target device (deviceAddr)
	2. Get the **number of blinks** or a command value (blinkCnt)

 char buf[100];

 sprintf(buf,"sending to device: %i\t number of blinks: %i",deviceAddr, blinkCnt);

 Serial.println(buf);

* This formats and prints the command being sent for debugging purposes.

 Wire.beginTransmission(deviceAddr); // start communication with the I2C device

 Wire.write(blinkCnt); // send the blink count as a byte

 Serial.println(Wire.endTransmission()); // ends transmission and prints the result

* Wire.beginTransmission(addr): Begins communication with a device at deviceAddr.
* Wire.write(blinkCnt): Sends the blink count to that device.
* Wire.endTransmission(): Ends the transmission. The return value indicates success (0) or various errors (1, 2, etc.), which is then printed to serial.

**🧠 Summary**

This Arduino code listens to the **Serial port**, expecting **two integers**:

1. An I2C device address
2. A value (e.g., number of blinks)

It then:

* Sends that value to the specified I2C address
* Prints what it sent and the result of the transmission

**🔧 Example**

If you send:

8 5

over the Serial Monitor, the Arduino will:

* Send 5 to the device at I2C address 8
* Print:
* sending to device: 8 number of blinks: 5
* 0

Let me know if you want help writing the code for the **receiver Arduino** that gets this data!

Sure! Let me explain what this Arduino code does, step by step:

**🔌 Libraries and Setup**

#include <Wire.h>

* Includes the **Wire** library, which allows the Arduino to communicate using **I2C** (a two-wire communication protocol).

void setup() {

 Serial.begin(115200);

 Wire.begin();

}

* Serial.begin(115200);: Initializes serial communication with the computer at 115200 bits per second.
* Wire.begin();: Initializes the Arduino as an **I2C master**, ready to send data over the I2C bus.

**🔁 Main Loop**

void loop() {

 while (Serial.available() > 0) {

 int deviceAddr = Serial.parseInt();

 int blinkCnt = Serial.parseInt();

* Serial.available() > 0: Checks if there is any data coming in from the serial port.
* Serial.parseInt(): Reads the next integer from the serial input. It’s used **twice** here to:
	1. Get the **I2C address** of the target device (deviceAddr)
	2. Get the **number of blinks** or a command value (blinkCnt)

 char buf[100];

 sprintf(buf,"sending to device: %i\t number of blinks: %i",deviceAddr, blinkCnt);

 Serial.println(buf);

* This formats and prints the command being sent for debugging purposes.

 Wire.beginTransmission(deviceAddr); // start communication with the I2C device

 Wire.write(blinkCnt); // send the blink count as a byte

 Serial.println(Wire.endTransmission()); // ends transmission and prints the result

* Wire.beginTransmission(addr): Begins communication with a device at deviceAddr.
* Wire.write(blinkCnt): Sends the blink count to that device.
* Wire.endTransmission(): Ends the transmission. The return value indicates success (0) or various errors (1, 2, etc.), which is then printed to serial.

**🧠 Summary**

This Arduino code listens to the **Serial port**, expecting **two integers**:

1. An I2C device address
2. A value (e.g., number of blinks)

It then:

* Sends that value to the specified I2C address
* Prints what it sent and the result of the transmission

**🔧 Example**

If you send:

8 5

over the Serial Monitor, the Arduino will:

* Send 5 to the device at I2C address 8
* Print:
* sending to device: 8 number of blinks: 5
* 0

Let me know if you want help writing the code for the **receiver Arduino** that gets this data!

Great! Now you're looking at the **I2C slave device** code — the one that receives the blink count from the master (which you shared earlier). Let me break it down clearly:

**🧠 Purpose**

This code sets up an Arduino as an **I2C slave** at address 9. When it receives a number from the master, it blinks the onboard LED (pin 13) that many times.

**🧱 Code Breakdown**

#define SLAVE\_ADDR 9

#include <Wire.h>

volatile int numOfBlink = 0;

* SLAVE\_ADDR 9: This device will listen for I2C messages addressed to **9**.
* volatile int numOfBlink: This is a shared variable that **can be modified inside an interrupt**, so it's declared volatile to avoid compiler optimizations that might break things.

**🧠 Forward Declaration**

void callback(int cnt);

* Declares a function that will be triggered whenever data is received over I2C.

**⚙️ Setup**

void setup() {

 Wire.begin(SLAVE\_ADDR); // Initialize this device as an I2C slave at address 9

 Serial.begin(115200); // Start Serial for debugging

 Wire.onReceive(callback); // Register the receive callback

 pinMode(13, 1); // Set pin 13 (built-in LED) as output

}

**📬 Receive Callback**

void callback(int cnt){

 numOfBlink = Wire.read();

 char buf[100];

 sprintf(buf,"I am device: %i\t number of blinks: %i",SLAVE\_ADDR, numOfBlink);

 Serial.println(buf);

}

* This function is called **automatically** whenever data is received via I2C.
* Wire.read() reads **one byte** from the master's message — the blink count.
* It stores it in numOfBlink, then prints a debug message to Serial.

**🔁 Main Loop**

void loop() {

 if(numOfBlink){

 interrupts(); // Ensure interrupts are enabled

 digitalWrite(13, 1); // Turn LED on

 delay(250);

 digitalWrite(13, 0); // Turn LED off

 delay(250);

 noInterrupts(); // Pause interrupts to safely update shared variable

 numOfBlink--;

 interrupts(); // Resume interrupts

 }

}

* If numOfBlink is greater than 0, it:
	+ Blinks the LED once (on for 250ms, off for 250ms)
	+ Decrements numOfBlink safely using noInterrupts() and interrupts() to prevent conflicts with the callback

**🧪 Example in Action**

If the master sends:

9 3

The slave device at address 9 will:

1. Receive the value 3 via I2C
2. Set numOfBlink = 3
3. Blink the LED on pin 13 three times

**✅ Summary**

| **Role** | **Function** |
| --- | --- |
| **Master** | Sends blink count to a slave using I2C |
| **Slave** | Listens for data, stores blink count, blinks |

Let me know if you'd like to expand this to more than one slave or send more complex data!