Electronic Basics #3: Programming an Attiny+Homemade Arduino Shield

ATtiny is a family of ultra-low-power microcontrollers from **Microchip Technology (formerly Atmel)**. These microcontrollers are part of the AVR architecture, offering a compact, cost-effective solution for embedded systems where small size, low power consumption, and minimal peripheral requirements are essential.

Popular ATtiny Models:

- **ATtiny85:**
    - **8 pins**
    - **8-bit, 5V operation, up to 20 MHz**
    - **8 KB Flash, 512 Bytes SRAM, 6 I/O pins**
- **ATtiny45:**
    - **8 pins**
    - **4.5V to 5.5V operation, 8 MHz clock**
    - **4 KB Flash, 256 Bytes SRAM, 5 I/O pins**
- **ATtiny2313:**
    - **20 pins**
    - **2 KB Flash, 128 Bytes SRAM, 16 I/O pins**

ATtiny85

ATtiny85 is a compact, low-power microcontroller from the **ATtiny** family, part of the **AVR architecture** developed by **Microchip Technology** (formerly Atmel). It's a popular choice in embedded systems, DIY projects, and applications where space and power efficiency are important.

Key Features of ATtiny85:

1. **Small Form Factor:**
    - The ATtiny85 is available in an **8-pin package**, making it ideal for projects with size constraints.
2. **Microcontroller Architecture:**
    - Based on the **AVR 8-bit architecture** with a **RISC (Reduced Instruction Set Computing)** core, offering efficient execution of commands.
3. **Memory:**
    - **8 KB Flash memory** for program storage.
    - **512 Bytes SRAM** for data storage.
    - **256 Bytes EEPROM** for non-volatile data storage.

4. **Clock Speed:**
    - The default internal clock speed is **8 MHz**, which can be adjusted via external oscillators if necessary, up to a maximum of **20 MHz**.
5. **I/O Pins:**
    - **6 I/O pins** available, which can be used for **digital input/output**. Some of these pins can also serve alternate functions like **PWM** and **Analog-to-Digital Conversion (ADC)**.
6. **Low Power Consumption:**
    - Designed for low-power applications, ATtiny85 offers multiple power-saving modes, such as **idle** and **standby**, allowing for efficient battery operation.
7. **Peripherals:**
    - **PWM outputs**: Useful for controlling motors, LEDs, and other devices that require variable power.
    - **Analog-to-Digital Converter (ADC)**: 4-channel, 10-bit ADC for reading analog signals (e.g., sensors).
    - **SPI**: Serial Peripheral Interface (SPI) for communication with other devices.
    - **Internal Timer**: For generating precise delays or PWM outputs.
8. **Programming:**
    - The ATtiny85 can be programmed using **Arduino IDE** with the help of an external programmer (e.g., **USBasp** or **UPDI** programmer).
    - It supports **Arduino bootloaders**, enabling easy use with the familiar Arduino environment.
9. **Wide Voltage Range:**
    - It operates in the range of **2.7V to 5.5V**, which makes it adaptable for both low-voltage and standard voltage applications.

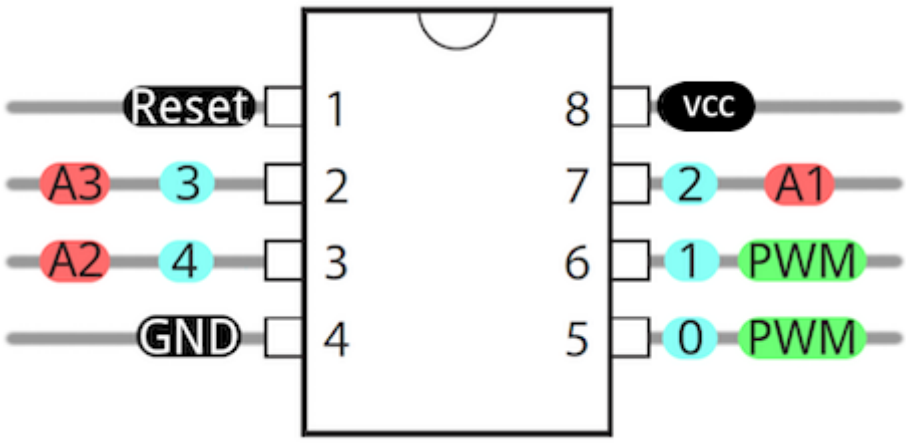

Fig:3.1 ATtiny 85 Microcontroller

Applications of ATtiny85:

1. **Wearable Electronics:**
    - Its small size and low power consumption make it suitable for wearable devices like fitness trackers and smartwatches.
2. **Battery-Powered Projects:**
    - Due to its low current consumption, ATtiny85 is ideal for battery-powered applications like sensors, remote controls, and wireless devices.
3. **LED and Motor Control:**
    - With **PWM outputs** and **ADC**, ATtiny85 is used for controlling LED brightness, motor speeds, and other analog components.
4. **Embedded Systems:**
    - Its small form factor and ease of integration in space-constrained systems make it an excellent choice for custom embedded designs.
5. **Hobby Projects:**
    - ATtiny85 is widely used in DIY electronics, such as custom circuits, robotics, and automation projects, due to its simplicity and Arduino compatibility.

Programming ATtiny85 with Arduino IDE:

To program ATtiny85 using the Arduino IDE, the following steps are generally required:

1. **Install ATtiny Core**: In the Arduino IDE, you must add the ATtiny85 board to the **Boards Manager**.
2. **Select Board and Programmer**: Choose **ATtiny85** under **Tools > Board** and set the programmer (e.g., **USBasp**) under **Tools > Programmer**.
3. **Upload Code**: Write and upload code using the Arduino IDE just like an Arduino board, but you will need an external programmer (USBasp or similar) to upload the code to the ATtiny85.

Example ATtiny85 Code (Blinking an LED):

```
// Pin for LED
const int ledPin = 0; // Pin 0 of ATtiny85

void setup() {
  pinMode(ledPin, OUTPUT); // Set LED pin as output
```

```
}

void loop() {
  digitalWrite(ledPin, HIGH);  // Turn LED ON
  delay(1000);                 // Wait for 1 second
  digitalWrite(ledPin, LOW);   // Turn LED OFF
  delay(1000);                 // Wait for 1 second
}
```

This code will blink an LED connected to **Pin 0** of the ATtiny85.