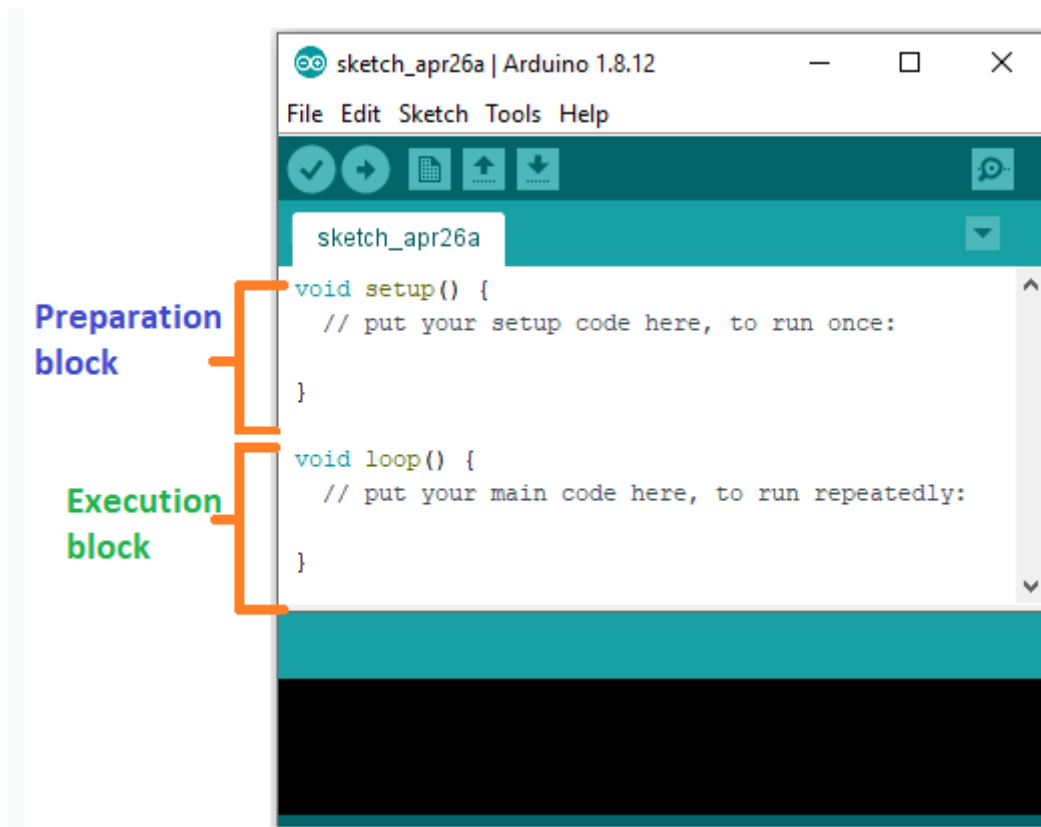


Arduino Programming:

कोडिंग स्क्रीन को दो ब्लॉक में बांटा गया है। सेटअप को प्रिपरेशन (तैयारी) ब्लॉक माना जाता है, जबकि लूप को execution (निष्पादन) ब्लॉक माना जाता है। इसे नीचे दिखाया गया है:



सेटअप और लूप ब्लॉक में स्टेटमेंट्स का सेट कर्ली ब्रैकेट्स के साथ संलग्न है। किसी विशेष प्रोजेक्ट के लिए कोडिंग आवश्यकताओं के आधार पर हम कई स्टेटमेंट लिख सकते हैं।

```
void setup ()  
{  
    Coding statement 1;  
    Coding statement 2;  
    .  
    .  
    .  
    Coding statement n;  
}
```

```

void loop ()
{
    Coding statement 1;
    Coding statement 2;
    .
    .
    .
    Coding statement n;
}

```

`setup()`: इसमें एकसिक्यूट किए जाने वाले कोड का प्रारंभिक भाग होता है। सेटअप सेक्शन में पिन मोड, लाइब्रेरी, वेरिएबल्स आदि को इनिशियलाइज़ किया जाता है। इसे केवल एक बार प्रोग्राम अपलोड करने के दौरान और Arduino बोर्ड को रीसेट या पावर अप करने के बाद निष्पादित किया जाता है।

शून्य सेटअप () प्रत्येक स्केच के शीर्ष पर रहता है। जैसे ही प्रोग्राम चलना शुरू होता है, `setup` फंक्शन के कर्ली ब्रैकेट के अंदर का कोड एकसिक्यूट होता है, और यह केवल एक बार ही एकसिक्यूट होता है।

`loop()` : लूप फंक्शन के कर्ली ब्रैकेट के अन्दर के स्टेटमेंट्स को बार बार एकसिक्यूट किया जाता है | वेरिएबल्स के वैल्यू के आधार पर `loop` फंक्शन के स्टेटमेंट्स को रिपीट किया जाता है |

Time in Arduino:

Arduino प्रोग्रामिंग में समय को एक मिलीसेकंड में मापा जाता है |

जहाँ, 1 सेकंड = 1000 मिलीसेकंड

हम मिलीसेकंड के अनुसार समय को समायोजित कर सकते हैं।

उदाहरण के लिए, 5-सेकंड की देरी के लिए, प्रदर्शित होने वाला समय 5000 मिलीसेकंड होगा।

Decision making Techniques:

डिसिशन मेकिंग स्टेटमेंट्स प्रोग्राम की दिशा और प्रवाह तय करते हैं | उन्हें कंडीशनल स्टेटमेंट्स के रूप में भी जाना जाता है क्योंकि वे बूलियन एक्सप्रेशन के साथ शर्तों को निर्दिष्ट करते हैं जिनका मूल्यांकन सही (true) या गलत (false) बूलियन मान के लिए किया जाता है |

डिसिशन मेकिंग स्टेटमेंट्स निम्नलिखित हैं :

If statement

If ... else statement

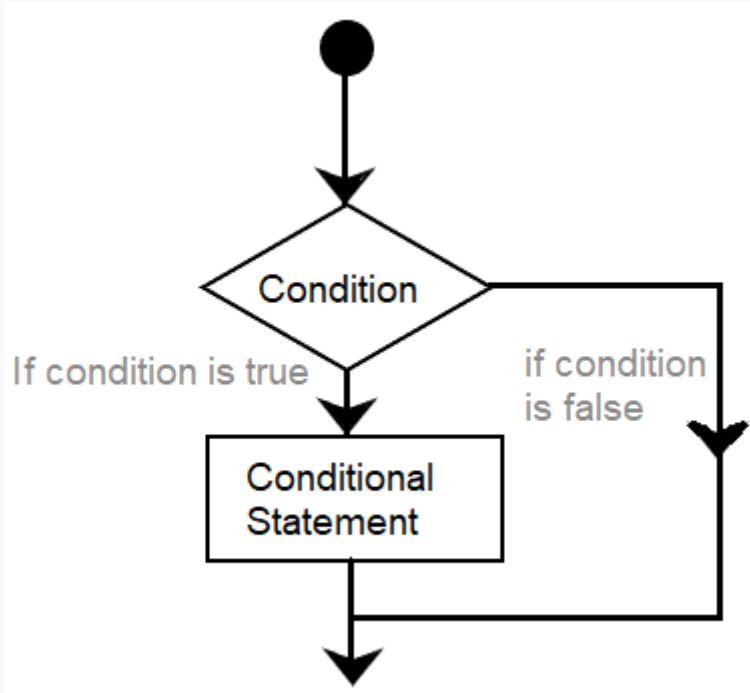
If ...else-if statement

Nested if statement

Switch statement

if statement:

यदि कोड में स्थिति सही है, तो संबंधित कार्य एक्सीक्यूट किया जाता है। if स्टेटमेंट स्थिति की जाँच करता है और फिर एक स्टेटमेंट या स्टेटमेंट का एक सेट निष्पादित करता है।



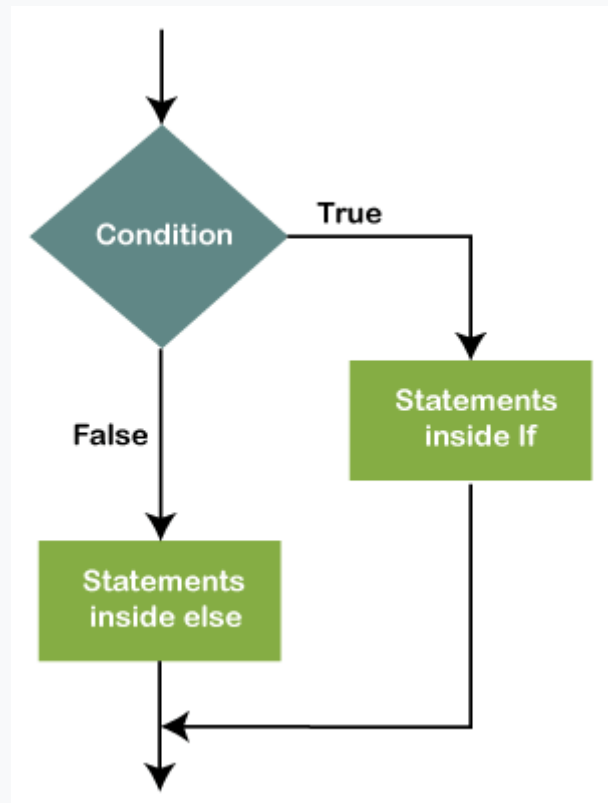
दिए गए फ्लोचार्ट में यह स्पष्ट है की यदि कंडीशन सही है तो स्टेटमेंट ब्लॉक एक्सीक्यूट होगा और यदि गलत है तो यह ब्लॉक एक्सीक्यूट नहीं होगा ।

If स्टेटमेंट का सिंटैक्स :

```
if ( condition)
{
// include statements
// if the condition is true
// then performs the function or task specified inside the curly braces
}
```

if else statement: if else स्टेटमेंट में if कंडीशन सही होने पर if ब्लॉक एक्सीक्यूट होता है एवं कंडीशन गलत होने पर else ब्लॉक एक्सीक्यूट होता है ।

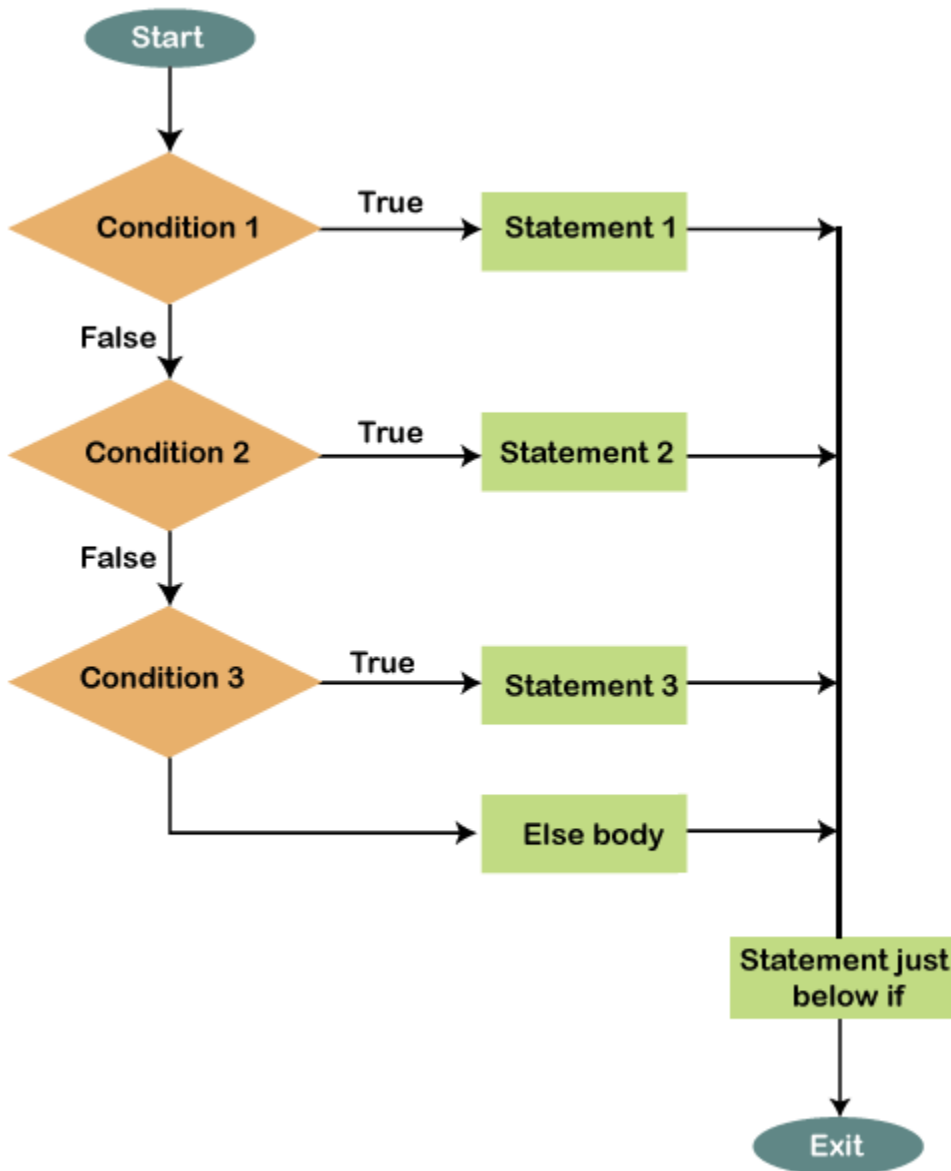
फ़्लोचार्ट नीचे दिखाया गया है:



If else सिंटेक्स:

```
if (condition)
{
// statements
}
else
{
//statements
}
```

else () कथन में अन्य if कथन भी शामिल हो सकते हैं। इसके कारण हम एक ही प्रोग्राम में कई Statement चला सकते हैं।



सही स्टेटमेंट मिलने तक स्टेटमेंट्स को एक-एक करके निष्पादित किया जाएगा। जब सही कथन मिल जाता है, तो यह कोड में अन्य सभी if और else कथनों को छोड़ देगा और कोड के संबंधित ब्लॉक को रन किया जायेगा ।

Looping Techniques

for Loop: लूप के लिए कर्ली ब्रैकेट के अंदर दिए गए बयानों को निर्दिष्ट स्थिति के अनुसार बार-बार निष्पादित किया जाता है। लूप के लिए इंक्रीमेंट काउंटर का उपयोग लूप रिपीटिशन को बढ़ाने या घटाने के लिए किया जाता है।

फॉर स्टेटमेंट का उपयोग आमतौर पर दोहराए जाने वाले कार्य या संचालन के लिए या सरणियों के संयोजन में डेटा/पिन के समूह पर काम करने के लिए किया जाता है।

for loop सिंटेक्स :

for (initialization; condition; increment)

```
{  
  \\ statements  
}
```

initialization: इसे वेरिएबल के आरंभीकरण के रूप में परिभाषित किया गया है।

condition: प्रत्येक निष्पादन पर स्थिति का परीक्षण किया जाता है। यदि स्थिति सही है, तो यह दिए गए कार्य को निष्पादित करेगा। लूप तभी समाप्त होता है जब कंडीशन गलत हो जाती है।

increment: इसमें इंक्रीमेंट ऑपरेटर शामिल है, जैसे कि $i++$, $i--$, $i+1$, इत्यादि। इसे हर बार तब तक बढ़ाया जाता है जब तक कि कंडीशन गलत न हो जाय |

उदाहरण के लिए :

1. **for** ($i = 0 ; i < 5 ; i + +$)

उपरोक्त कथन लूप को पांच बार निष्पादित करेगा। i का मान 0 से 4 तक होगा।

2. **for** ($i = 0 ; i <= 5 ; i + +$)

उपरोक्त कथन लूप को छह बार निष्पादित करेगा। i का मान 0 से 5 तक होगा।

while loop : while लूप कंडीशनल लूप है जो कोष्ठक के अंदर कोड को निष्पादित करना जारी रखता है जब तक कि निर्दिष्ट स्थिति गलत नहीं हो जाती।

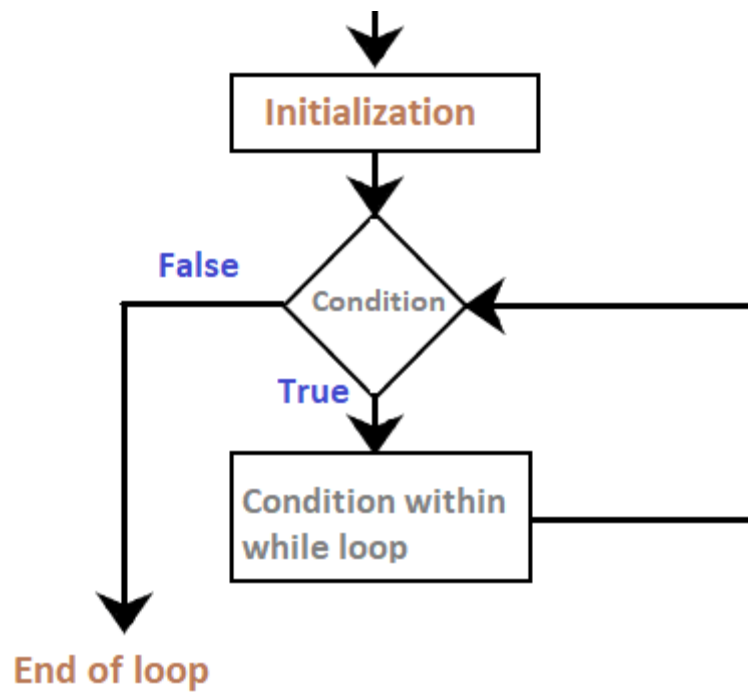
परीक्षण की स्थिति को बदलने या रोकने के लिए बनाए जाने तक while लूप कभी बाहर नहीं निकलेगा। Arduino में while लूप के सामान्य उपयोग में सेंसर परीक्षण, कैलिब्रेशन (सेंसर के इनपुट को कैलिब्रेट करना), वेरिएबल इंक्रीमेंट आदि शामिल हैं।

While loop सिंटैक्स :

```
while (condition)  
{  
  // code or set of statements  
}
```

कंडीशन: यह बूलियन एक्सप्रेशन को निर्दिष्ट करता है, जो कंडीशन के सही या गलत होने का निर्धारण करता है।

While loop फ्लोचार्ट :



Do-while loop:

Do-while लूप की कार्यप्रणाली, while लूप के समान है। do-while के अंदर की कंडीशन कम से कम एक बार एक्जीक्यूट होगी। ऐसा इसलिए है क्योंकि शुरुआत के बजाय लूप के अंत में स्थिति का परीक्षण किया जाता है।

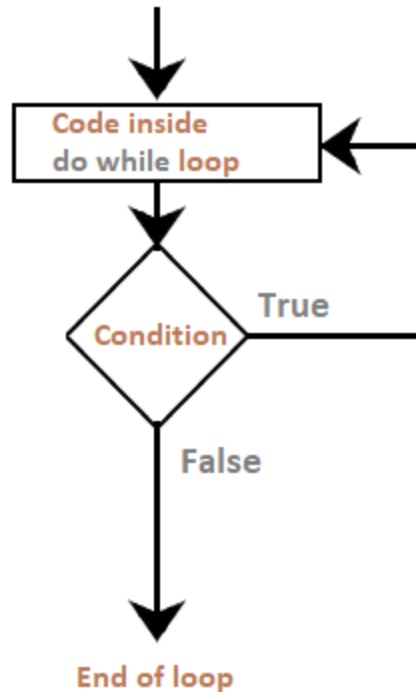
Do-while syntax:

```

do
{
// code or set of statements
} while (condition);
  
```

कंडीशन: यह बूलियन एक्सप्रेशन को निर्दिष्ट करता है, जो कंडीशन के सही या गलत होने का निर्धारण करता है।

Flowchart of do-while loop



Concept of Input and Output port of embedded development

board:

Arduino बोर्ड पर पिन को इनपुट या आउटपुट के रूप में कॉन्फ़िगर किया जा सकता है। यह ध्यान रखना महत्वपूर्ण है कि अधिकांश Arduino एनालॉग पिन, डिजिटल पिन के समान ही कॉन्फ़िगर और उपयोग किए जा सकते हैं।

Pins Configured as INPUT:

Arduino पिन डिफ़ॉल्ट रूप से इनपुट के रूप में कॉन्फ़िगर किए गए हैं, इसलिए जब आप उन्हें इनपुट के रूप में उपयोग कर रहे हों तो उन्हें pinMode() के साथ इनपुट के रूप में स्पष्ट रूप से घोषित करने की आवश्यकता नहीं है। इस तरह से कॉन्फ़िगर किए गए पिन को उच्च-प्रतिबाधा (high impedance) अवस्था में कहा जाता है।

इनपुट पिन के सामने 100 मेगाओम के श्रृंखला प्रतिरोधी के बराबर प्रतिरोध होने के कारण इनपुट पिन सैंपलिंग सर्किट पर बहुत कम मांग करते हैं |

इसका मतलब है कि इनपुट पिन को एक स्टेट से दूसरे स्टेट में स्विच करने में बहुत कम करंट लगता है। यह कैपेसिटिव टच सेंसर को लागू करने या एक एलईडी को फोटोडायोड के रूप में पढ़ने जैसे कार्यों के लिए पिन को उपयोगी बनाता है।

Pins Configured as OUTPUT:

pinMode() के साथ OUTPUT के रूप में कॉन्फ़िगर किए गए पिन को कम-प्रतिबाधा अवस्था में कहा जाता है। इसका मतलब है कि वे अन्य सर्किटों को पर्याप्त मात्रा में करंट प्रदान कर सकते हैं। एटमेगा पिन अन्य उपकरणों/सर्किट को 40 mA (मिलीएम्पस) करंट स्रोत की तरह सकारात्मक विद्युत् प्रवाह प्रदान कर सकते हैं या सिंक की तरह नकारात्मक करंट प्रदान कर सकते हैं। यह एक एलईडी को उज्ज्वल रूप से रोशन करने के लिए या कई सेंसर को चलाने के लिए पर्याप्त करंट है, लेकिन रिले, सोलनॉइड्स या मोटर्स को चलाने के लिए पर्याप्त करंट नहीं है।

आउटपुट पिन से उच्च धारा उपकरणों को चलाने का प्रयास, पिन में आउटपुट ट्रांजिस्टर को नुकसान पहुंचा सकता है या नष्ट कर सकता है, या संपूर्ण एटमेगा चिप को नुकसान पहुंचा सकता है। अक्सर, इसके परिणामस्वरूप माइक्रोकंट्रोलर में "dead" पिन हो सकता है लेकिन शेष चिप्स अभी भी पर्याप्त रूप से कार्य करते हैं। इस कारण से, 470Ω या 1k रेसिस्टर्स के माध्यम से OUTPUT पिन को अन्य उपकरणों से कनेक्ट करना एक अच्छा विचार है, जब तक कि किसी विशेष एप्लिकेशन के लिए पिन से अधिकतम करंट की आवश्यकता न हो।

pinMode() Function:

pinMode() फ़ंक्शन का उपयोग एक विशिष्ट पिन को इनपुट या आउटपुट के रूप में कॉन्फ़िगर करने के लिए किया जाता है। INPUT_PULLUP मोड के साथ आंतरिक पुल-अप प्रतिरोधों को सक्षम करना संभव है। इसके अतिरिक्त, INPUT मोड आंतरिक पुल-अप को स्पष्ट रूप से अक्षम करता है।

pinMode function का syntax:

```
pinMode (pin, mode)
```

जहाँ, pin : पिन की संख्या जिसका मोड आप सेट करना चाहते हैं |

mode: हम संबंधित पिन नंबर के अनुसार मोड को INPUT या OUTPUT के रूप में सेट कर सकते हैं।

उदाहरण:

```

int button = 5 ; // button connected to pin 5
int LED = 6; // LED connected to pin 6

void setup () {
  pinMode(button , INPUT_PULLUP);
  // set the digital pin as input with pull-up resistor
  pinMode(button , OUTPUT); // set the digital pin as output
}

void setup () {
  If (digitalRead(button ) == LOW) // if button pressed {
    digitalWrite(LED,HIGH); // turn on led
    delay(500); // delay for 500 ms
    digitalWrite(LED,LOW); // turn off led
    delay(500); // delay for 500 ms
  }
}

```

digitalWrite() Function:

DigitalWrite () फ़ंक्शन का उपयोग डिजिटल पिन पर उच्च या निम्न मान लिखने के लिए किया जाता है। यदि पिन को pinMode() के साथ एक OUTPUT के रूप में कॉन्फ़िगर किया गया है, तो इसका वोल्टेज हाई के लिए 5V (या 3.3V बोर्ड पर 3.3V) और LOW के लिए 0V (ग्राउंड) पर सेट किया जाएगा। यदि पिन को INPUT के रूप में कॉन्फ़िगर किया गया है, तो digitalWrite() इनपुट पिन पर आंतरिक पुलअप को इनेबल (HIGH) या डिसेबल (LOW) करेगा। इंटरनल पुलअप रेसिस्टर को इनेबल करने के लिए pinMode() को INPUT_PULLUP पर सेट करने की अनुशंसा की जाती है।

यदि आप pinMode() को OUTPUT पर सेट नहीं करते हैं, और एक एलईडी को उस पिन से कनेक्ट करते हैं, तो digitalWrite(HIGH) कॉल करते समय, एलईडी मंद दिखाई दे सकती है। pinMode() को स्पष्ट रूप से सेट किए बिना, digitalWrite() ने आंतरिक पुल-अप प्रतिरोधी को इनेबल किया होगा, जो एक बड़े विद्युत्-सीमित प्रतिरोधी (करंट लिमिटिंग रेसिस्टर) की तरह कार्य करता है।

digitalWrite() फ़ंक्शन का सिंटैक्स :

```

void loop() {
  digitalWrite (pin ,value);
}

```

pin : पिन की संख्या जिसका मोड आप सेट करना चाहते हैं |
value: HIGH या LOW

उदाहरण:

```
int LED = 6; // LED connected to pin 6

void setup () {
  pinMode(LED, OUTPUT); // set the digital pin as output
}

void loop () {
  digitalWrite(LED,HIGH); // turn on led
  delay(500); // delay for 500 ms
  digitalWrite(LED,LOW); // turn off led
  delay(500); // delay for 500 ms
}
```

analogRead() function:

Arduino, digitalRead() फ़ंक्शन के माध्यम से, यह पता लगा सकता है कि क्या उसके किसी एक पिन पर वोल्टेज लगाया गया है। एक ऑन/ऑफ सेंसर (जो किसी वस्तु की उपस्थिति का पता लगाता है) और एक एनालॉग सेंसर जिसका मूल्य लगातार बदलता रहता है, के बीच अंतर होता है। इस प्रकार के एनालॉग सेंसर को पढ़ने के लिए हमें एक अलग प्रकार के पिन की आवश्यकता होती है।

Arduino बोर्ड के निचले-दाएँ भाग में, आपको "एनालॉग इन" के रूप में चिन्हित छह पिन दिखाई देंगे। ये विशेष पिन न केवल यह बताते हैं कि उन पर कोई वोल्टेज लगाया गया है, बल्कि इसका मूल्य भी बताते हैं। analogRead() फ़ंक्शन का उपयोग करके, हम किसी एक पिन पर लागू वोल्टेज को पढ़ सकते हैं।

यह फ़ंक्शन 0 और 1023 के बीच की संख्या लौटाता है, जो 0 और 5 वोल्ट के बीच वोल्टेज का प्रतिनिधित्व करता है। उदाहरण के लिए, यदि पिन नंबर 0 पर 2.5 V का वोल्टेज लगाया जाता है, तो AnalogRead(0) 512 मान देता है।

analogRead() function का सिंटैक्स :

```
analogRead(pin);
```

pin: एनालॉग इनपुट पिन की संख्या (ज्यादातर बोर्ड पर 0 से 5, मिनी और नैनो पर 0 से 7, मेगा पर 0 से 15)

उदाहरण:

```
int analogPin = 3; //potentiometer wiper (middle terminal)
    // connected to analog pin 3
int val = 0; // variable to store the value read

void setup() {
    Serial.begin(9600); // setup serial
}

void loop() {
    val = analogRead(analogPin); // read the input pin
    Serial.println(val); // debug value
}
```

Interface serial port with Embedded development Board:

Serial communication in Arduino:

Arduino में, "सीरियल कम्युनिकेशन" का अर्थ श्रृंखला में डेटा को किसी अन्य डिवाइस में स्थानांतरित करना है। Arduino में, हम Arduino के USB प्लग और TX/RX पिन के माध्यम से कंप्यूटर या कुछ अन्य उपकरणों के साथ सीरियल संचार कर सकते हैं। Arduino में सीरियल संचार उन पिनो के माध्यम से किया जाता है जो इस उद्देश्य के लिए समर्पित हैं। सीरियल संचार डेटा के प्रत्येक बाइट को अन्य डिवाइस या कंप्यूटर पर स्थानांतरित करना सुनिश्चित करता है ।

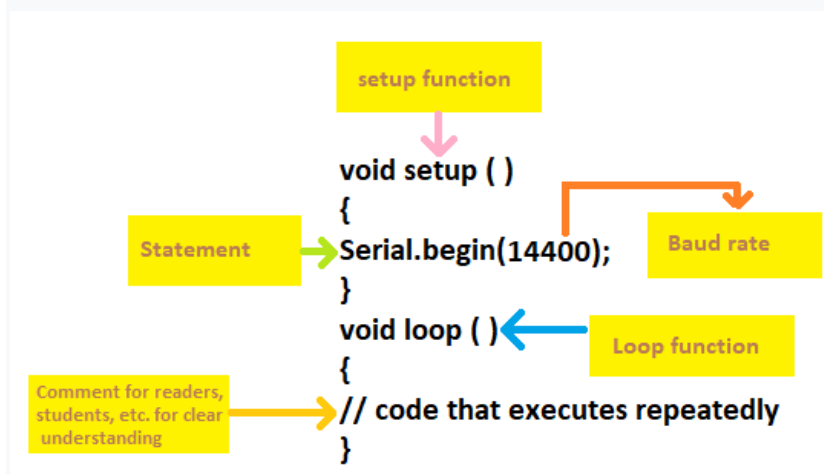
Arduino Uno में, दो पिन; पिन 0 और पिन 1 को UART (यूनिवर्सल एसिंक्रोनस रिसीवर ट्रांसमीटर) और USART (यूनिवर्सल सिंक्रोनस एसिंक्रोनस रिसीवर ट्रांसमीटर) के रूप में जाने जाने वाले सीरियल कम्युनिकेशन के लिए असाइन किया गया है और उन्हें Tx/Rx पिन के रूप में भी जाना जाता है। ये पिन 3.3 वोल्ट या 5 वोल्ट पर संचालित होते हैं, इसलिए उन्हें RS232 सीरियल पोर्ट से जोड़ने की अनुशंसा नहीं की जाती है क्योंकि यह 12 वोल्ट पर संचालित होता है जो Arduino बोर्ड को नुकसान पहुंचा सकता है, इसके अलावा, सीरियल कम्युनिकेशन यूएसबी प्लग की सहायता से कंप्यूटर के माध्यम से भी किया जा सकता है।

सीरियल संचार का आउटपुट सीरियल मॉनिटर पर देखा जा सकता है, जिसे टूल के ड्रॉप-डाउन मेनू में "सीरियल मॉनिटर" पर क्लिक करके "Arduino IDE" में एक्सेस किया जा सकता है। कंप्यूटर के साथ सीरियल संचार के लिए, Arduino को USB केबल के माध्यम से कंप्यूटर से कनेक्ट करें |

Arduino के विभिन्न बिल्ट-इन फंक्शन हैं लेकिन सीरियल संचार के लिए सबसे अधिक उपयोग किये जाने वाले फंक्शन निम्नलिखित हैं:

Serial.begin(speed) : इस फंक्शन का उपयोग विशिष्ट बॉड दर पर डेटा स्थानांतरित करने की गति को सेट करने के लिए किया जाता है

Arduino में डिफॉल्ट बॉड दर 9600 बीपीएस (बिट्स प्रति सेकंड) है। हम अन्य बॉड दरों को भी निर्दिष्ट कर सकते हैं, जैसे कि 4800, 14400, 38400, 28800, आदि।



Serial.read() : यह फंक्शन दूसरे कनेक्टेड मशीन से Arduino में आने वाले सीरियल डेटा को पढ़ता है। यहाँ int डेटा प्रकार का उपयोग किया जाता है। यह आने वाले सीरियल डेटा का पहला डेटा बाइट लौटाता है। सीरियल पोर्ट पर कोई डेटा उपलब्ध नहीं होने पर यह -1 भी लौटाता है।

Serial.print() : यह फंक्शन डेटा को ASCII में परिवर्तित करता है जो मनुष्यों द्वारा आसानी से पढ़ा जा सकता है और इसे सीरियल मॉनिटर पर प्रिंट करता है।

Example 1:

1. `Serial.print(15.452732)`

Output: 15.45

यह प्रिंटर को सिंगल कैरेक्टर के रूप में बाइट भेजता है। Arduino में, Serial.print() का उपयोग करने वाले स्ट्रिंग्स और कैरेक्टर जैसे हैं वैसे ही भेजे जाते हैं।

Example 2:

1. Serial.print("Hello Arduino")

Output: "Hello Arduino"

Serial.println(): यह फ़ंक्शन प्रिंट () के समान काम करता है, लेकिन इसके अलावा, यह एक नई लाइन जोड़ता है।

Serial.flush(): यह फ़ंक्शन आउटगोइंग सीरियल डेटा के प्रसारण को पूरा करना सुनिश्चित करता है।

उदाहरण:

```
void setup ()
{
  Serial.begin ( 4800);
}
void loop ()
{
  Serial.print(" Hello");
  delay(1000);
  Serial.println("Arduino");
  delay ( 1500);
}
```

हम Arduino के USB प्लग के माध्यम से कंप्यूटर के साथ सीरियल संचार के लिए Serial.begin() फ़ंक्शन का उपयोग करेंगे, और डेटा को 4800 बॉड दर पर स्थानांतरित करने की गति निर्धारित करेंगे। फिर हम सीरियल मॉनिटर पर "Hello" टेक्स्ट को प्रिंट करने के लिए Serial.print("Hello") फ़ंक्शन का उपयोग करेंगे, फिर 1.5 सेकंड्स का डिले के लिये delay() फ़ंक्शन का उपयोग करेंगे | Serial.println("Arduino") फ़ंक्शन "Arduino" print करने के बाद एक नया लाइन जोड़ेगा, जब loop फ़ंक्शन जब अगली बार एकसीक्यूट होगा तो "Hello Arduino" नये लाइन में print होगा |

Make a basic circuit of embedded development board:

Blinking an LED:

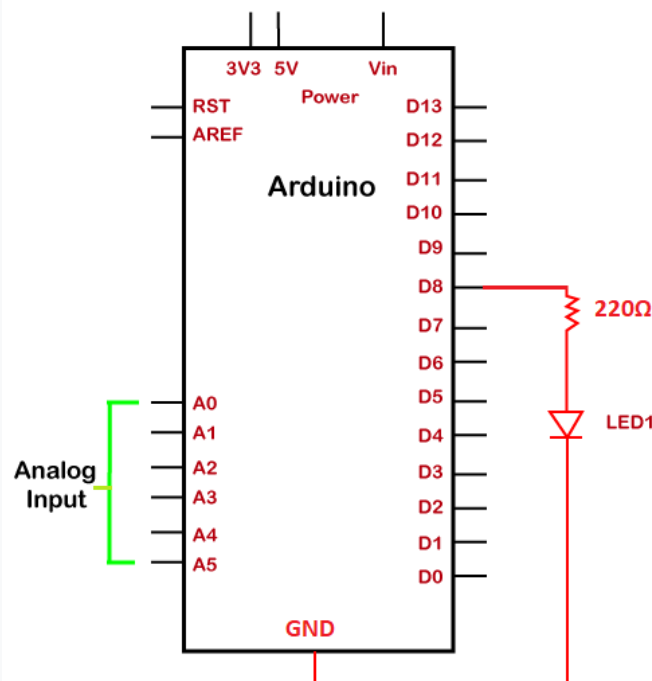
यह Arduino का उपयोग करके बनाई गई सरल बेसिक प्रोजेक्ट है। एलईडी (लाइट एमिटिंग डायोड) एक इलेक्ट्रॉनिक उपकरण है, जो अपने टर्मिनलों से करंट गुजरने पर प्रकाश उत्सर्जित करता है। एलईडी का उपयोग विभिन्न अनुप्रयोगों में किया जाता है। इसका उपयोग विभिन्न इलेक्ट्रॉनिक उपकरणों में ON/OFF संकेतक के रूप में भी किया जाता है।

इस परियोजना में, हम Arduino बोर्ड पर LED को डिजिटल पिन से जोड़ेंगे। एलईडी एक साधारण प्रकाश के रूप में काम करेगा जिसे निर्दिष्ट अवधि के लिए चालू और बंद किया जा सकता है।

एलईडी के ब्लिंकिंग में उपयोग किए जाने वाले घटक नीचे सूचीबद्ध हैं:

1. 1x Arduino UNO board.
2. 1 x Breadboard
3. 2 x Jump wires
4. 1 x LED
5. 1 x Resistor of 220 Ohm

हम 470 ओम तक के किसी भी मान के प्रतिरोधक का उपयोग कर सकते हैं। हम अपनी सर्किट आवश्यकताओं के आधार पर प्रतिरोधकों के अन्य मूल्यों का भी उपयोग कर सकते हैं। आमतौर पर, मान स्वीकार्य फॉरवर्ड करंट से अधिक नहीं होना चाहिए।



सर्किट डायग्राम स्पष्ट रूप से UNO बोर्ड के पिनआउट को दर्शाती है। यह बोर्ड से जुड़े एलईडी और प्रतिरोध को भी प्रदर्शित करता है।

LED ब्लिंकिंग के लिए प्रोग्राम :

```
void setup ()
{
  pinMode ( 8, OUTPUT); // to set the OUTPUT mode of pin number 8.
}
void loop ()
{
  digitalWrite (8, HIGH);
  delay(1000); // 1 second = 1 x 1000 milliseconds
  digitalWrite (8, LOW);
  delay(500); // 0.5 second = 0.5 x 1000 milliseconds
}
```

Explain interfacing DC Motor with programming

डीसी मोटर को सबसे सरल मोटर माना जाता है, जिसमें घरों से लेकर उद्योगों तक के विभिन्न अनुप्रयोग होते हैं। उदाहरण में कार, इलेक्ट्रिक वाहनों का इलेक्ट्रिक विंडो, लिफ्ट आदि शामिल है।

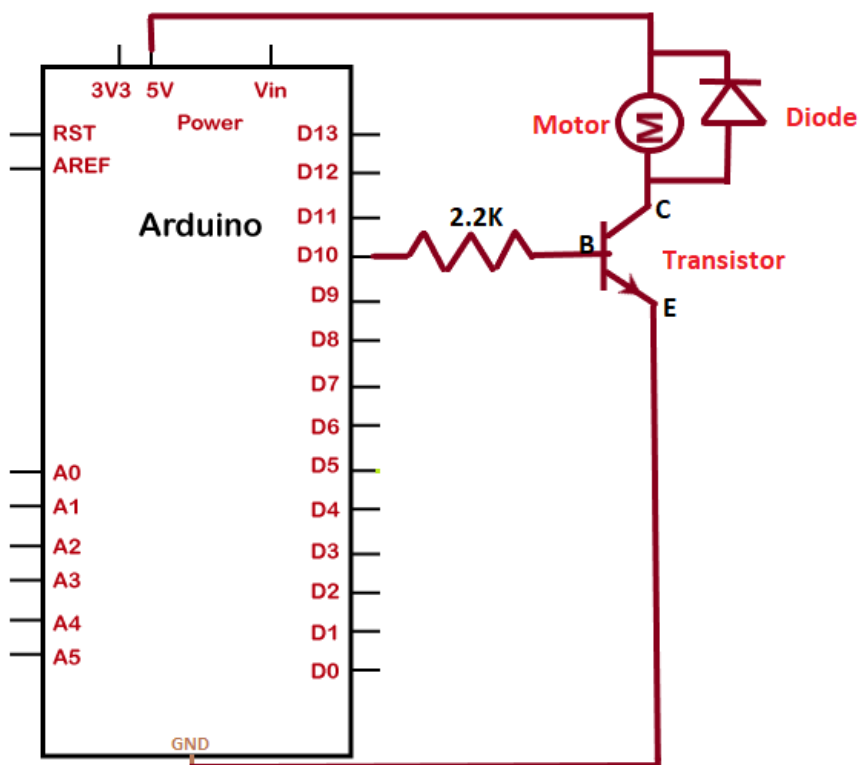
डीसी मोटर्स का सिद्धांत विद्युत चुम्बकीय प्रेरण पर आधारित है। इसका अर्थ है कि मोटर का घूर्णन चुम्बकीय क्षेत्र द्वारा उत्पन्न बल पर निर्भर करता है। यह विद्युत ऊर्जा को यांत्रिक ऊर्जा में परिवर्तित करता है। इन मोटरों को डायरेक्ट करंट से संचालित किया जा सकता है।

डायोड, ट्रांजिस्टर और रेसिस्टर का उपयोग करके Arduino बोर्ड के साथ DC मोटर का सरल कनेक्शन:

कनेक्शन के लिए आवश्यक कंपोनेंट्स :

- Arduino UNO R3 board
- Breadboard
- A Resistor of 2.2K Ohm
- Transistor (NPN)
- Diode
- DC Motor

- Jump wires



```
int PinOFmotor = 10;
// PIN 10 of the Arduino is initialized to the variable
// the pin must be a PWM pin
void setup()
{
  pinMode(PinOFmotor, OUTPUT);
}
void loop()
{
  digitalWrite(PinOFmotor, HIGH);
  delay(1000);
  digitalWrite(PinOFmotor, LOW);
  delay(1000);
}
```

कनेक्शन स्थापित करने के चरण नीचे सूचीबद्ध हैं:

1. Arduino बोर्ड के 10 (PWM) पिन करने के लिए रेसिस्टर के एक छोर को कनेक्ट करें।
2. रेसिस्टर के दूसरे छोर को ट्रांजिस्टर के मध्य पिन से कनेक्ट करें।
3. ट्रांजिस्टर के एक सिरे के टर्मिनल को Arduino के GND पिन से और दूसरे सिरे के टर्मिनल को डायोड से कनेक्ट करें।
4. डायोड के बैंड फेसिंग टर्मिनल को Arduino बोर्ड के 5V पिन से कनेक्ट करें।
5. डीसी मोटर के एक सिरे वाले टर्मिनल को डायोड के बैंड फेसिंग टर्मिनल से कनेक्ट करें।
6. डीसी मोटर के दूसरे सिरे के टर्मिनल को डायोड के दूसरे सिरे से जोड़ें।

बोर्ड में कोड अपलोड करने के चरण:

1. Arduino IDE खोलें।
2. Tools -> Board -> Arduino UNO से बोर्ड के प्रकार का चयन करें।
3. Tools -> Port -> COM से पोर्ट का चयन करें।
4. स्केच को कनेक्शन आरेख में अपलोड करें