Manual dos Blocos de Controle – Ardublock

Nome do Bloco, para que serve e exemplos de aplicação.

* “Sempre”



Figura 1 – Bloco “sempre”(loop)

Este é o bloco principal do Ardublock, com ele damos início à nossa programação e todos os outros blocos que estiverem dentro dele sempre irão ficar rodando, no caso um loop infinito.

**Exemplo:** Na *Figura 2.a* temos um bloco LED que está ligado dentro do bloco “sempre” e ele permanecerá assim até que o arduino seja desligado.

E na *Figura 2.b* temos nossa programação no arduino, onde automaticamente dentro do “void setup ( ) ” ele declara se o pino que estamos utilizando para conectar o LED no AttoBox é uma entrada ou uma saída*,* neste caso ele indica que é uma saída “OUTPUT”, no próximo bloco falaremos um pouco mais sobre a função do void setup (). E dentro do “void loop ( ) ” está contido toda a programação que colocamos no bloco “sempre”.



Figura 2.a – Exemplo com um LED em loop



Figura 3 – Exemplo da programação no Arduino

* “Programa”

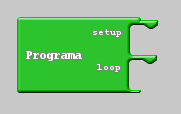


Figura 4 – Bloco de Programa com “setup”

Este bloco tem a mesma função que o bloco anterior, que é de dar início à nossa programação e também ficar em loop infinito, mas o diferencial é que temos, além do “Loop”, o “Setup” neste bloco. O setup é chamado no momento que o Arduino é ligado, e é usado para inicializar variáveis, definir os modos de entrada ou saída dos pinos, indicar bibliotecas entre outras. Essa função é executada apenas uma vez, quando o Arduino é iniciado ou reiniciado (um botão no arduino que quando apertado reinicia suas configurações e inicia o arduino novamente). Use o “Setup” para comandos devem ser executados apenas uma vez na inicialização.

OBS: Para que o Ardublock funcione corretamente é obrigatório ter apenas um bloco de inicialização na tela, por isso tem que escolher entre o bloco de “Setup” ou o bloco de “Programa”

**Exemplo:** No bloco anterior “*Sempre*” falamos um pouco sobre a função “*void setup ( )*” que apareceu no nosso código do arduino, onde automaticamente ele declarava se o pino do LED que está conectado na AttoBox é uma entrada ou uma saída. Agora nesse novo bloco, nós mesmo podemos declarar novas variáveis conforme a necessidade de programação. Na imagem abaixo *Figura 4*, colocamos na opção “Setup” o componente LED conectado na porta D13 ligado e dentro do “Loop” arrumamos um outro bloco de LED conectado na porta D12 e um bloco“delay” de 1000 milissegundos e com isso o LED irá ficar piscando.

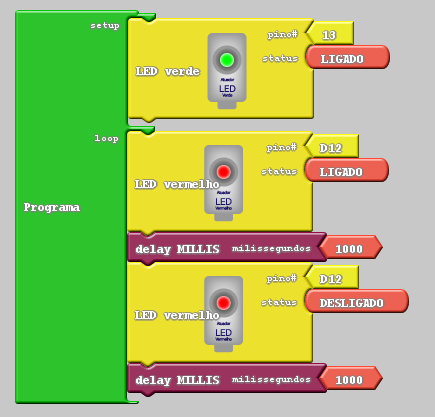


Figura 5 – Exemplo do bloco “Programa” e do bloco “delay MILLIS”

* “Delay MILLIS”



Figura 6 – delay Milissegundos

O bloco Delay Milissegundos pausa o programa por uma quantidade especificada de tempo (em milissegundos). Cada 1000 milissegundos equivalem a 1 segundo.

Exemplo: Na *Figura 4* o exemplo que utilizamos é de um LED conectado na porta D12 que inicia ligado, e depois nosso bloco delay, onde fica por 1000 milissegundos e muda o estado desse LED para desligado, assim sendo, espera mais 1000 milissegundos e inicia o código novamente mudando o estado do LED para ligado.

* “Delay MICROS”

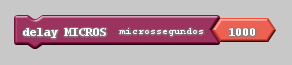


Figura 7 - delay Microssegundos

O bloco Delay Microssegundos pausa o programa pela quantidade de tempo especificada como parâmetro (em microssegundos). Há mil microssegundos em um milissegundo, e um milhão de microssegundos em um segundo. Atualmente, o maior valor que irá produzir um delay preciso é 16383. Isso pode mudar em versões futuras do Arduino. Para delays mais longos que alguns milhares de microssegundos, você deve usar “delay Milissegundos” em vez disso.

**Exemplo:** Na *Figura 8* o exemplo que utilizamos é de um LED que inicia ligado, e depois nosso bloco delay, onde fica por 16383 microssegundos e muda o estado desse LED para desligado, assim sendo, espera mais 16383 microssegundos e inicia o código novamente mudando o estado do LED para ligado. Nesse caso ver o LED piscando é quase impossível, devido ao tempo que ele liga e desliga ser muito rápido.

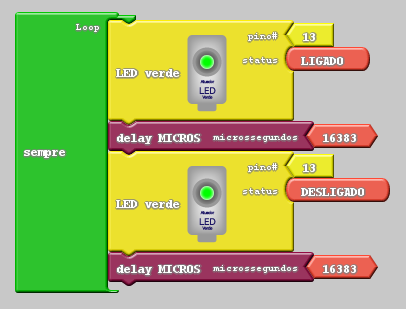


Figura 8 - Exemplo da aplicação do Delay Microssegundos

* tempo (ms)



Figura 9 - tempo (ms)

Este bloco retorna o número de milissegundos passados desde que a placa Arduino começou a executar o programa atual. Esse número irá sofrer overflow (chegar ao maior número possível e então voltar para zero), após aproximadamente 50 dias.

Exemplo: Neste exemplo o código lê e imprime o tempo em milissegundos desde que a AttoBox foi ligada. Colocamos um Delay Milissegundos para não enviar uma quantidade massiva de dados na tela para o usuário.

Falaremos um pouco mais sobre os blocos *“imprime”* e *“junte”* no guia de blocos de comunicação.



Figura 10 - Exemplo da aplicação do bloco Tempo(ms)

* “Se” ou no Inglês “if”



Figura 11 - bloco "se"

O bloco *“se”* ou *“if”* checa uma condição ou faz um teste se a condição seja verdadeira e caso a mesma corresponda que sim, ele executa o código estabelecido dentro do *“então”.*

**Exemplo:** Na figura abaixo *Figura 12* precisamos entender que o nosso bloco *“se”* temos a função *“teste”* onde encontramos uma comparação entre um Sensor Touch conectado na porta 10 do nosso AttoBox, e o estado que ele se encontra, se Falso ou Verdadeiro.

Neste caso quando pressionamos o Sensor Touch o seu estado é verdadeiro, fazendo a comparação dentro do *“teste”* e essa afirmação de fato ocorre, ele irá executar o que está dentro do *“então”*, sendo assim ligando nosso LED que está conectado na porta 13 da AttoBox.

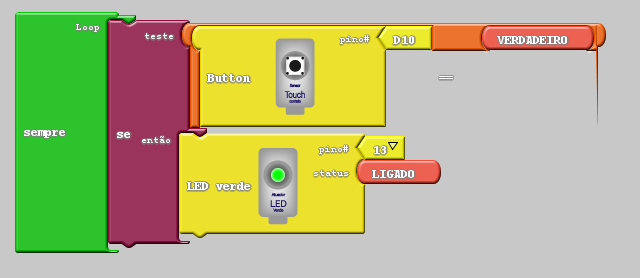


Figura 12 – Exemplo de aplicação do bloco “Se”

* “Se/Senão” ou “if/else”



Figura 13 – Bloco se/senão

O bloco se/senão ou if/else é parecido com o bloco anterior “se” mas tem uma vantagem que é a opção do senão, quando o programa está rodando e chega nesse bloco, a primeira coisa ele vai fazer o “teste” comparação das variáveis para ver se é verdadeiro ou falso, caso seja verdadeiro ele entra na opção “então”, mas caso nosso teste de falso, ele vai para a opção “senão” e roda o que estiver conectado ali dentro. No exemplo abaixo irá ficar mais claro o seu funcionamento.

**Exemplo:** Como já falamos no bloco anterior, quando pressionado o botão seu estado é verdadeiro, então irá executar o que está dentro do “então”, que neste caso é um LED conectado na porta 13 com o estado de ligado, mas quando soltamos o botão seu estado é alterado para falso, então a condição de comparação é falsa e a partir disso, ele irá executar o que está dentro da parte “senão”, sendo assim, irá desligar o LED que está conectado na porta 13.

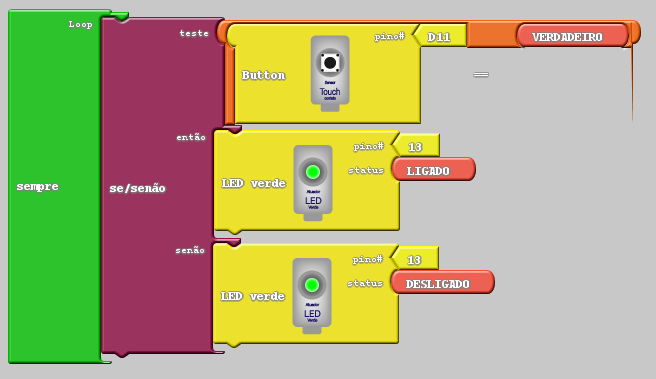


Figura 14 – Exemplo de Aplicação do Bloco “se/senão”

* “enquanto” ou “while”

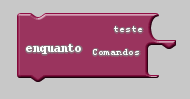


Figura 15 – Bloco “enquanto”

No bloco enquanto, os blocos que estiverem dentro do “Comandos” irão executar infinitamente enquanto a expressão que estiver dentro do “teste” seja verdadeira. Caso o estado da expressão dentro do “teste” se torne falsa, ele irá sair do loop e executar o restante do código.

Exemplo: podemos ver na figura abaixo o nosso bloco “enquanto”, e dentro da opção “teste” temos a comparação da variável “Atto” onde ela inicia com o valor zero automaticamente, e o teste é feito se ela é menor que cinco (5) ou não. Nesse caso a variável Atto é menor que 5 então irá executar o que está dentro da opção “Comandos” uma vez e depois que terminar tudo o que está ali dentro, ele volta para comparar novamente, sendo verdade ele repete o programa novamente, até a expressão “Atto < 5” se tornar falsa, que é quando nossa variável atingir o valor de 5 ou mais, e com isso ela acaba saindo do bloco “enquanto” e indo para o restante do programa contido dentro do bloco loop.

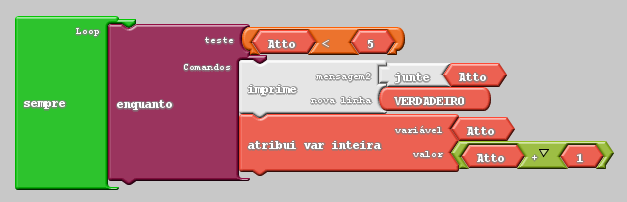


Figura 16 – Exemplo de aplicação do bloco enquanto

* “faça enquanto” ou “do-while”

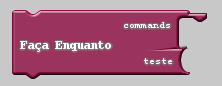


Figura 17 – Bloco Faça Enquanto

Esse bloco tem praticamente a mesma função que o bloco anterior, só que sua diferença se faz no momento que ocorre a comparação da variável, o anterior ele primeiro faz a comparação se é verdadeira e depois executa. Já esse aqui, primeiro executa uma vez o código que está dentro do “comando” e depois ele verifica se a comparação de variável é verdadeira, caso seja afirmativa ele executa novamente e caso seja falsa, ele pula para o restante da programação.

Exemplo: Como citado anteriormente, este bloco irá executar o programa que está dentro da opção “comando” e depois verificar se a expressão é verdadeira ou falsa. Neste exemplo observamos que ele já inicia apresentando o valor contido na variável “Atto” e já soma +1 nessa mesma variável, só depois ele compara para ver se a variável “Atto” é menor que 5, se for menor, ele executa mais uma vez e assim por diante até o valor chegar em 5, a comparação vai ser feita e como a variável “Atto” não é menor que 5, ele irá sairá do bloco “Faça Enquanto” e continuará no bloco loop, mas um detalhe importante, é que no bloco que comentamos anteriormente, ele mostrava os números até 4, esse já vai mostrar o número 5 porque só depois de executar todo o programa que vai existir a comparação.

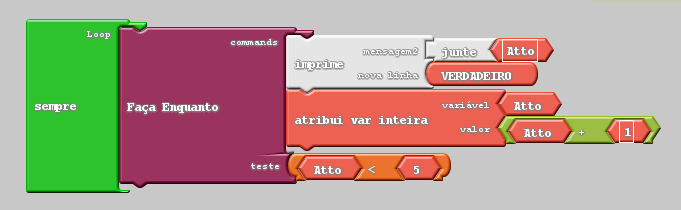


Figura 18 – Exemplo de Aplicação do bloco “Faça Enquanto”

* “Repita” ou “for”



Figura 19 – Bloco “repita”

O bloco “Repita” ou “for” é usado para repetir determinadas vezes os blocos envolvidos dentro do encaixe “Comandos”. Este bloco é útil para qualquer operação repetitiva. Logo no inicio ele testa a condição, no caso se a quantidade de vezes repetida ainda e menor que a desejada, e assim sendo verdadeira, os blocos de comando são executados, quando a condição for falsa, ele termina essa execução e continua executando o bloco “sempre” até iniciar novamente e chegar no bloco “repita” onde irá executar todo o processo novamente.

Exemplo: Neste exemplo colocamos no nosso bloco uma repetição de 5 vezes, e no encaixe “Comandos” acoplamos um led ligado e ele permanecerá neste estado por 1 segundo, depois desse 1 segundo, o seu estado muda para desligado onde fica por mais 1 segundo. Esse seria uma repetição das cinco vezes que vai acontecer, e depois de acontecer essas cinco vezes, no caso o ledo piscar cinco vezes com intervalos de 1 segundo, o programa sai do bloco repita e executa nosso bloco delay por 5 segundos, e volta para o início do bloco “loop” e irá fazer toda a execução citada anteriormente.

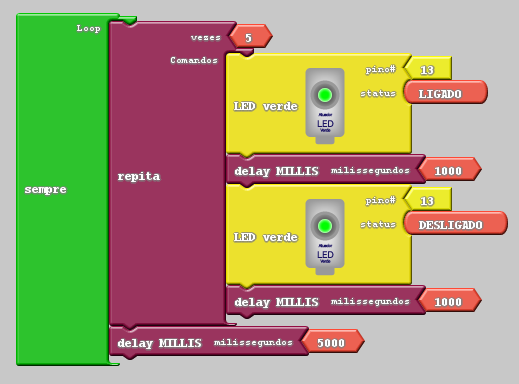


Figura 20 – Exemplo de Aplicação do bloco “repita”

* “Repete e conta”

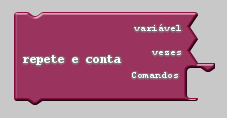


Figura 21 – Bloco repete e conta

Neste bloco o funcionamento é praticamente igual o anterior que era o apenas “repete”, mas o diferencial dele é que temo um encaixe “variável” que por padrão vem um bloco com o nome de uma variável, no exemplo mostraremos melhor para que fique mais claro, mas por padrão ele sempre tem valor inicial de 1, e irá sendo incrementado a cada vez que nosso bloco for repetindo até atingir o valor de vezes que foi digitado pelo usuário no encaixe “vezes”.

Exemplo: Na Figura 22 podemos observar que no encaixe “variável” tempo um bloco chamado (variable name) como mencionado na descrição a cima, seu padrão é iniciar com valor 1, e a cada repetição seu valor é incrementado em +1, então neste exemplo, dentro do encaixe “Comandos” foi acoplado um bloco que vai mostrar o valor dessa variável (variable name) a cada 1 segundo, e depois que terminar, irá aparecer a mensagem de (Fim do programa), ele espera 5 segundos e inicia as repetições novamente.

Na Figura 23 podemos analisar a mensagem descrita onde diz (Valor da Variável) e o valor seguinte é devido nosso bloco (variable name), então ele executa 5 vezes e depois aparece a mensagem de Fim do programa.

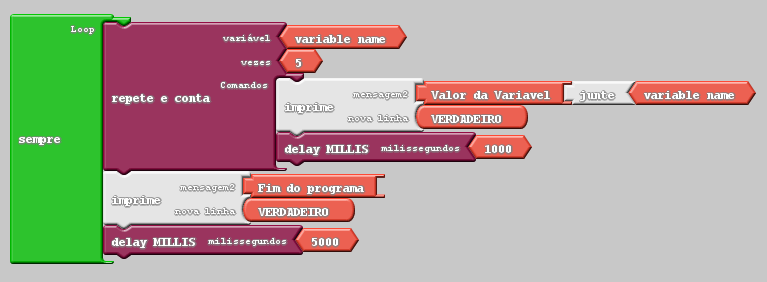


Figura 22 – Exemplo de Aplicação do bloco repete e conta

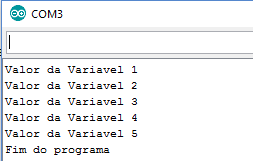


Figura 23 – Aplicação do bloco em funcionamento

* “Repita entre”



Figura 24 – Bloco Repita entre

O bloco repita entre, vai ficar repetindo o que está dentro do encaixe “commands”, muito parecido com os dois blocos anteriores de repetição, mas nesse podemos escolher qual o valor de início da minha variável, a quantidade de números por vez que ela vai pular, e até qual o valor final que vai ser atingido.

Exemplo: Neste exemplo explicaremos um pouco melhor o funcionamento deste bloco, iniciando pelo encaixe “variável” onde colocamos um nome qualquer desejável a sua escolha, aqui colocamos o nome de “Atto”, e após isso no encaixe “início” arrumamos o valor inicial para a nossa variável Atto, sendo que na figura 25 esse valor é 1. O próximo encaixe é o “Parar” e refere-se qual será o valor final de repetições que nosso programa vai poder chegar, no caso ele pode ir de 1 até 10. Mas temos um detalhe importante neste bloco, que é o encaixe “step of”, e ele representa a quantidade de números que ele pula a cada vez que acontece uma repetição, na figura 26 fica mais claro o que estamos fazendo, a variável Atto inicia em 1, e a cada repetição ele pula 2 números chegando até 9. Você pode perguntar, mas não arrumamos 10 como valor final? Sim, mas quando ele chegou em 9, ele pulou 2 números e foi direto para 11, o nosso bloco fez a verificação e viu que a variável Atto era maior que 10, então saiu deste bloco e continuou na programação do bloco “Sempre”.



Figura 25 - Exemplo de aplicação do bloco repita entre

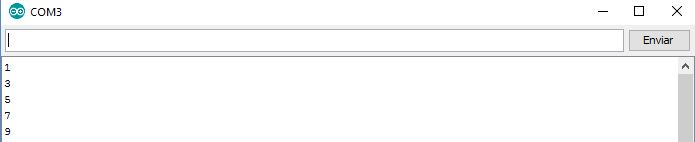


Figura 26 - Execução da aplicação do bloco

* “Pare”



Figura 27 – Bloco Pare

O bloco pare e usado para sair dos blocos de repetição, por exemplo o “repetir”, “enquanto” entre outros citados anteriormente, ele ignora a condição padrão das repetições. Sendo assim, quando chegar neste bloco ele sai do bloco repetição e continua normalmente o restante dos blocos que estão dentro do bloco “sempre”.

Exemplo: Analisando a Figura 28, temos um bloco de repetição, que irá repetir 5 vezes o que estiver dentro dele a princípio, mas no encaixe Comandos, temos o bloco “se/senão” onde já comentamos sobre ele no início deste manual, e sabemos que quando nossa variável “Atto” for menor que 2 ele irá fazer o que está no encaixe “senão” que no caso, ele repete e conta o número 1 e o número 2, depois quando nossa repetição chega em 3, o teste se a variável Atto é maior que dois se torna verdadeira e imprime na tela a mensagem “Sair do Loop” e pula para o bloco “pare”, e sua função é encerrar nosso programa de repetições, então ele não conta até 5 que nem seria o esperado. Sendo assim, quando chega no pare ele sai do bloco repete e conta e continua com o restante da programação que estará dentro do bloco “sempre”.

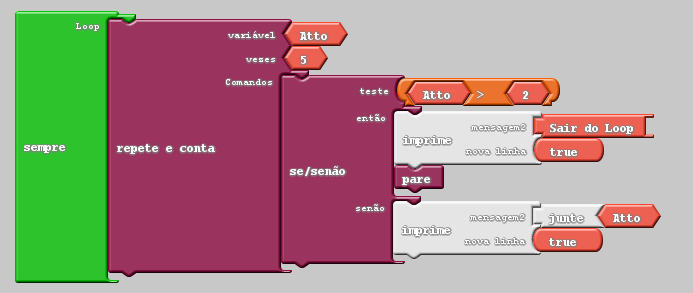


Figura 28 – Exemplo da aplicação do bloco Pare

* Subrotina



Figura 29 – Bloco Subrotina



Figura 30 – Bloco Subrotina

Aqui vamos apresentar dois blocos, porque um depende do outro para ser executado. O primeiro da Figura 29, é o bloco que irá criar uma função, uma subtorina, isso quer dizer um conjunto de comandos que realiza uma tarefa específica em um módulo dependente do código.

Já a Figura 30, é arrumada dentro do bloco “sempre” e quando executada, ela chama nossa função da figura 29, executando tudo o que estiver ali dentro. Nesse caso, o bloco da figura 30 serve para que sempre que a gente queira chamar a função ou a programação que estiver dentro do bloco da figura 29, usamos esta subrotina.

Exemplo: No exemplo da Figura 31 estamos usando outros blocos que já aprendemos neste manual, o repete e conta, e o se/senão. Sabemos que primeiro nosso programa irá executar 10 vezes o que está dentro do bloco repete e conta, e dentro dele, temos um bloco de se/senão, onde nossa comparação é se a variável Atto é menor que 5 ou não, quando a expressão for verdadeira, ela entra no encaixe “então” e executa a subrotina1, que é o mesmo bloco da figura 30, quando solicitada, ela executa tudo o que estiver dentro do bloco subrotina apresentado na figura 29, e imprime na tela “Subrotina 1” e o valor da variável Atto. E quando nosso teste se torna falso no bloco se/senão, ele vai para a subrotina2, e executa tudo o que está dentro do outro bloco de subrotina.

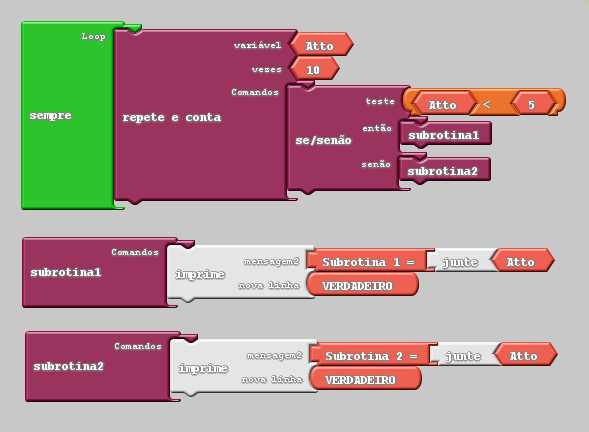


Figura 31 – Aplicação dos Blocos de Subrotina