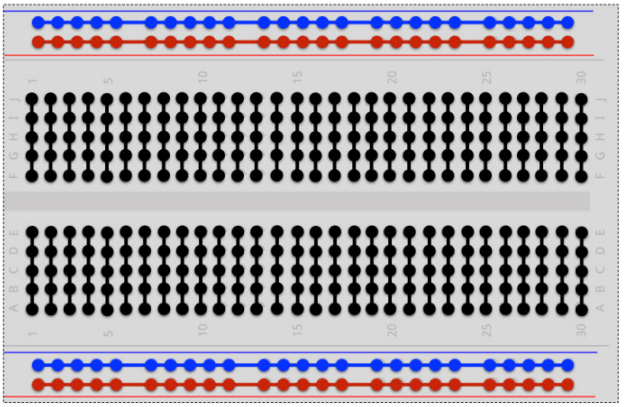
Empleando sensores y actuadores

Hasta ahora hemos trabajado con elementos internos de la placa, sin embargo gran parte del potencial de Arduino se encuentra en su capacidad de recoger datos y actuar sobre el medio. A continuación explicaremos la placa donde montaremos los circuitos y los distintos pines de la placa, que permiten la comunicación entre el programa y el circuito.

Protoboard

Para poder montar circuitos emplearemos una “protoboard” o placa de montaje como la siguiente donde cada punto representa una conexión y las líneas unen aquellas que se encuentran conectadas entre ellas. De forma que podemos separar entre dos tipos de conexiones:



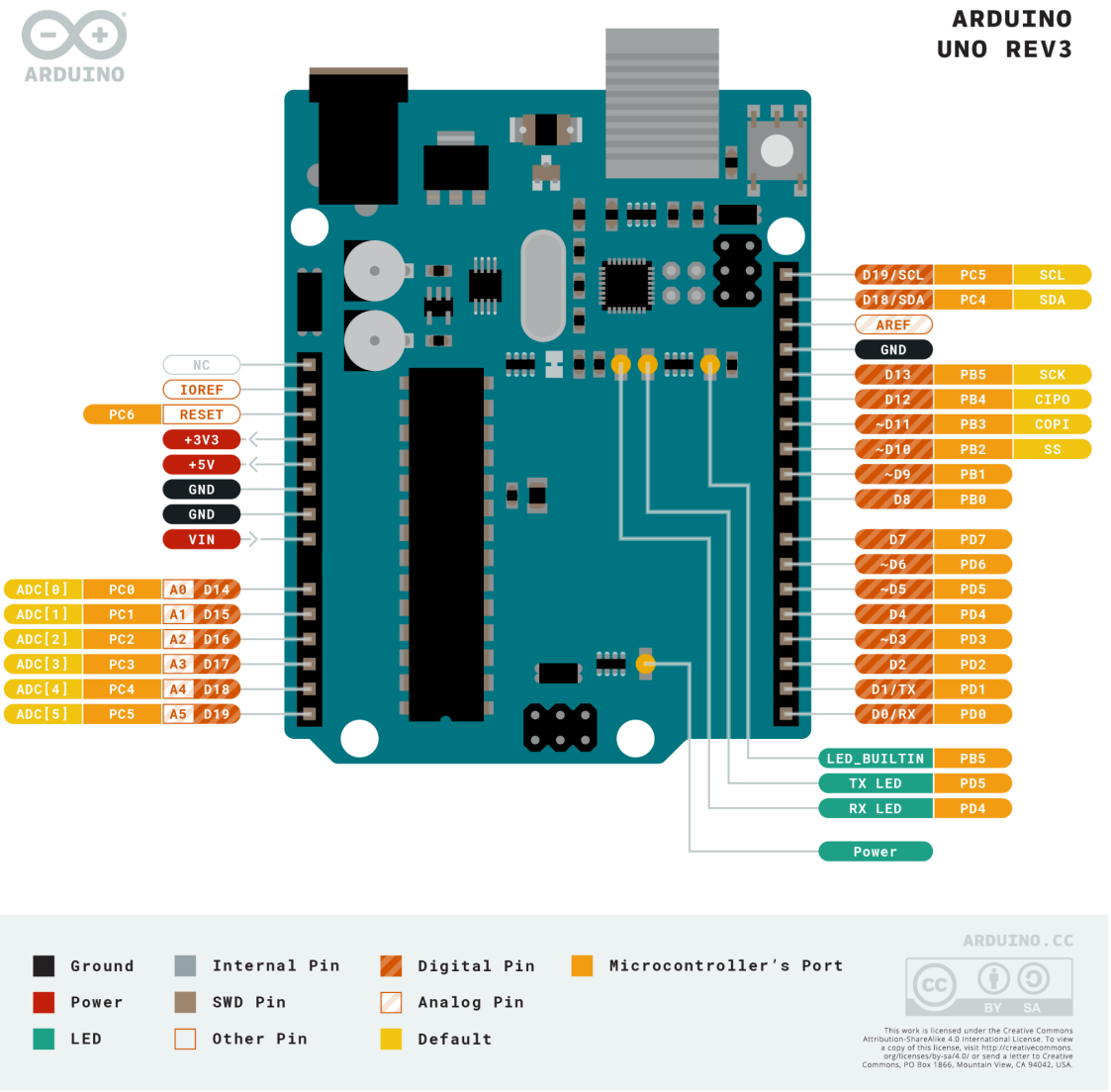
- **Los raíles de alimentación,** que siguen las líneas azules y rojas en grupos de 5 conexiones y se encuentran conectados de forma horizontal se emplean para dar corriente a los circuitos. Por ejemplo, las conexiones rojas superiores se encuentran todas en contacto entre ellas, pero aisladas de cualquier otra conexión, como el raíl azul superior o cualquiera de los raíles inferiores.
- **Los raíles de conexión,** en color negro y conexiones verticales, se emplean para colocar componentes como resistencias, LEDs o botones que aprenderemos a usar ahora y configurar su comunicación con la tabla a través de las distintas entradas y salidas de nuestra placa Arduino. Por ejemplo si deseamos conectar una resistencia en serie con un botón, como en el próximo proyecto, debemos conectar uno de los extremos de la resistencia en la misma línea vertical que la “patita” del botón.

Pines de la placa

La comunicación de la placa con el entorno se realiza a través de sus pines y conexiones:

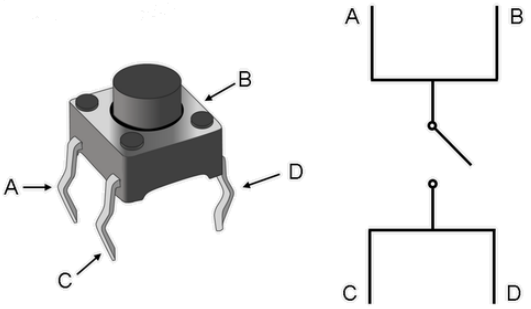
- **Pines digitales**: Digital Pins | Arduino Documentation, pueden configurarse tanto como entrada (para leer, sensores) o como salida (para escribir, actuadores). Las señales digitales son las que únicamente tienen dos estados, 0 o 1, es decir, 0 o 5V.
- **Pines analógicos de entrada**: Analog Input Pins | Arduino Documentation, usan un conversor analógico/digital y sirven para leer sensores analógicos como sondas de temperatura esto significa que pueden adoptar cualquier valor entre 0 y 1023.

- **Pines analógicos de salida (PWM)**: [Basics of PWM (Pulse Width Modulation) | Arduino Documentation], la mayoría de Arduino no tienen conversor digital/analógico y para tener salidas analógicas se usa la técnica PWM. No todos los pines digitales soportan PWM debemos fijarnos en aquellos que lleven el símbolo “~”, sus valores pueden ir desde 0 a 255.
- **Puertos de comunicación:** USB, serie, I2C y SPI



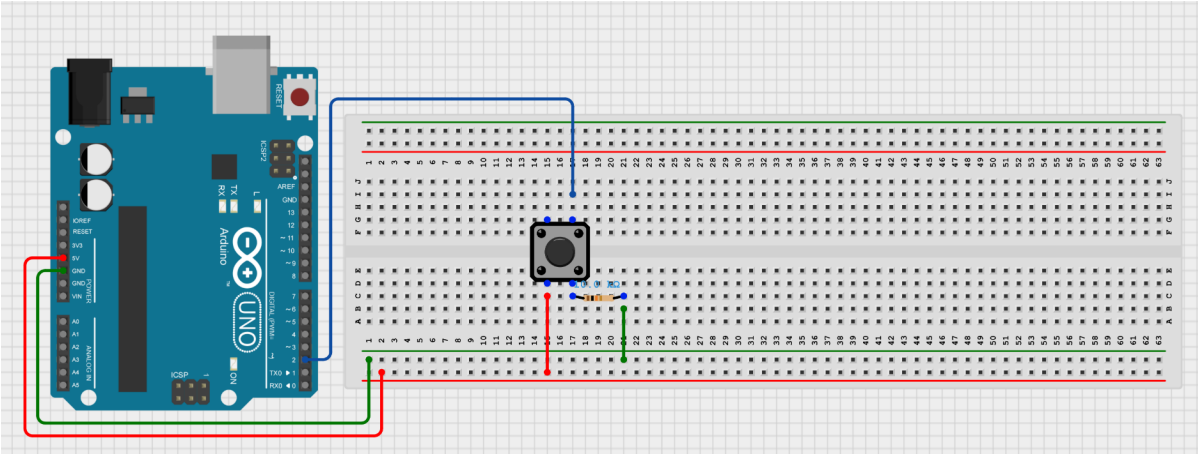
Utilizando un botón

En el proyecto inicial se conmuta el LED interno de la placa cada segundo, sin embargo a continuación aprenderemos a realizar esa conmutación pulsando un botón: Como podemos observar en la imagen, los pines A y B, así como C y D están conectados de forma que podemos controlar dos señales con el mismo pulsador.



Montaje

Para que la placa pueda leer las entradas que le mandemos debemos conectar el botón a la placa mediante uno de los pines digitales, por ejemplo el 2:



Cabe destacar que se utiliza una resistencia de 10k como pullup por seguridad y para evitar interferencias.

Programa

Ahora que hemos conectado el botón a nuestra placa deberemos cambiar el código para que el LED se active cuando estemos pulsando el botón. Para ello declararemos como constante el pin al que lo hemos conectado (el 2), estableceremos el pin como entrada y cambiaremos la lógica que se sigue para encender el LED.

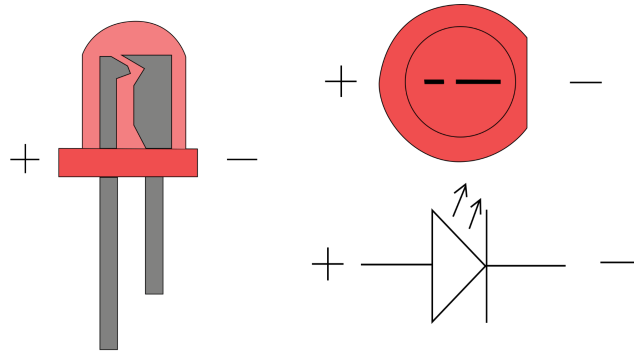
```
int buttonPin = 2;
int buttonState = 0;

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(buttonPin, INPUT);
}

void loop() {
  buttonState = digitalRead(buttonPin);
```

```
  if (buttonState == HIGH) {
      digitalWrite(LED_BUILTIN, HIGH);
  } else if (buttonState == LOW) {
      digitalWrite(LED_BUILTIN, LOW);
  }
  delay(50);
}
```

Proponemos como ejercicio modificar el código para, con el mismo montaje, obtener el funcionamiento inverso y que el botón apague la luz en lugar de encenderla. (Pista: intercambiar **algunos** valores de HIGH y LOW en el loop())

Utilizando un LED externo

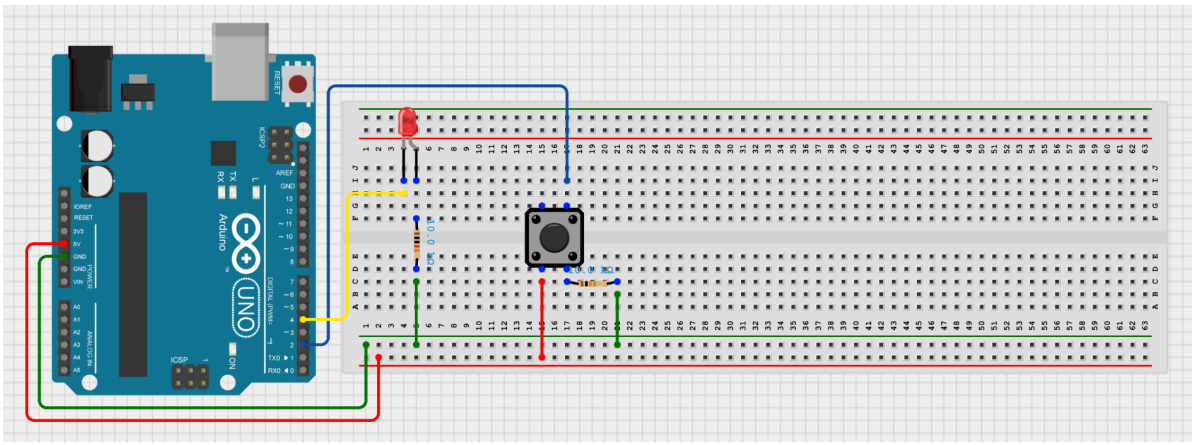
Ahora emplearemos un LED externo en lugar del interno, para ello simplemente tendremos que conectarlo y alterar ligeramente el código, aunque antes de conectarlo es importante destacar que los diodos LED (Light Emitting Diode), como un diodo estándar, sólo conducen en una dirección, de forma que el signo negativo debe de ir conectado a tierra (GND)



podemos diferenciar el lado negativo fijándonos en el extremo más corto o mirando desde arriba el diodo y fijándonos en el lado plano

Montaje

Para emplear un LED simplemente tendremos que conectarlo junto con una resistencia de 10k (para evitar cortocircuitos) a cualquier pin (el 4 por ejemplo):



Programa

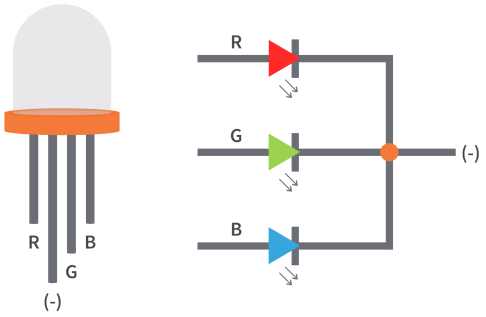
Modificando el código anterior, simplemente creamos una constante con el pin del LED y sustituimos todos los `LED_BUILTIN` por esta nueva constante:

```
int buttonPin = 2;
int ledPin = 4;

int buttonState = 0;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT);
}

void loop() {
  buttonState = digitalRead(buttonPin);
  if (buttonState == HIGH) {
      digitalWrite(ledPin, HIGH);
  } else if (buttonState == LOW) {
      digitalWrite(ledPin, LOW);
  }
  delay(50);
}
```

Potenciómetro y diodo RGB

Ahora que hemos aprendido el funcionamiento de un diodo estándar, proponemos el montaje de un proyecto un poco más complejo, vamos a regular el color de un diodo RGB mediante un potenciómetro, es decir una entrada analógica y tres salidas “analógicas”.

Diodo RGB

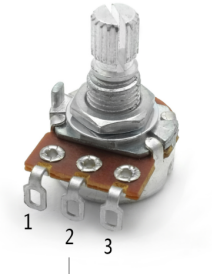


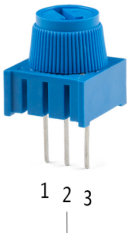
Estos diodos son ligeramente más complejos que los que hemos empleado en el ejemplo anterior ya que cuentan con tres diodos de los simples, que comparten su conexión negativa por lo que nos quedamos con cuatro conexiones, una para el positivo de cada uno de los colores y una en común para todos los negativos.

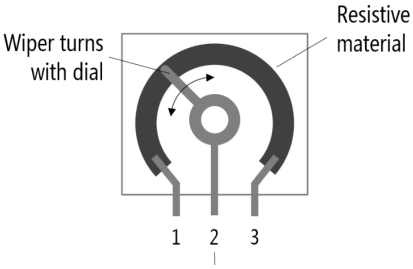
Para regular el color que emite el diodo, debemos cambiar la intensidad que atraviesa cada uno de los colores, realizando combinaciones entre los tres colores “Red, Green, Blue” (RGB) es decir rojo, verde y azul. Para poder variar el color, debemos variar la intensidad, y para esto no podemos emplear una señal digital como antes (ON-OFF) por lo que emplearemos una señal “analógica”, realmente es un pulso digital (PWM) que simula una señal analógica que nos permite dar valores con distinta intensidad a nuestro diodo y que se indican en la placa con el símbolo ~ detrás del número.

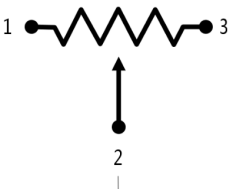
Potenciómetro

El potenciómetro no es más que una resistencia (como las que hemos empleado hasta ahora) que puede variar su valor con una ruleta, esto nos puede permitir variar de forma manual señales lo cual nos puede servir, como en este caso, para regular otro tipo de salidas variables. Funcionan con un cursor que gira sobre un material resistivo actuando como divisor de tensión, conectamos un extremo a la corriente y el otro a tierra y obtenemos en valor variable en la conexión del medio.



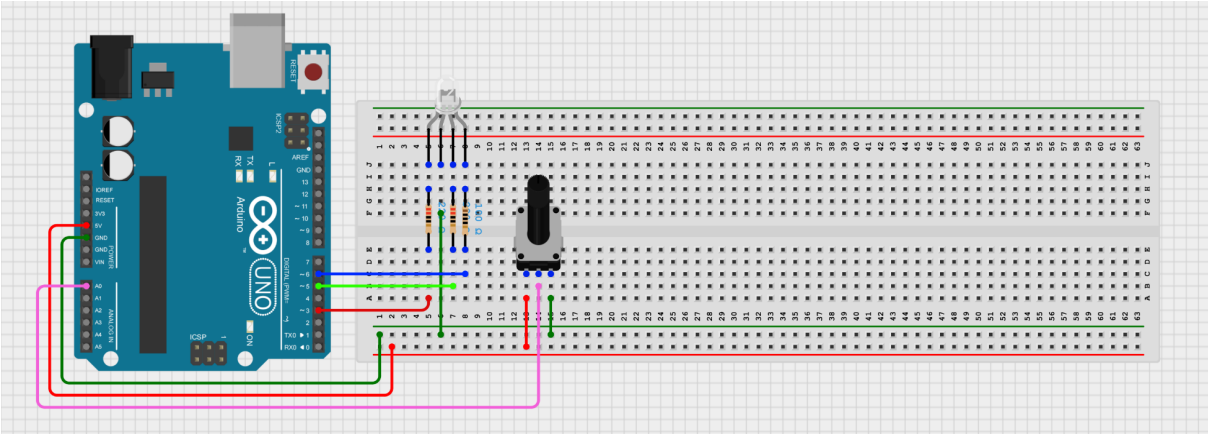






Montaje

Igual que en el ejemplo previo, vamos a proteger los diodos con resistencias, en este caso vamos a emplear dos resistencias de 220Ω y una de 100Ω, esto se debe a que el color azul se percibe peor por el ojo humano y el propio diodo necesita un poco más de tensión para funcionar y de esta forma quedan visualmente más igualados los colores.



Como hemos explicado hasta ahora, empleamos los raíles horizontales para alimentar el sistema conectándolo a 5V y GND.

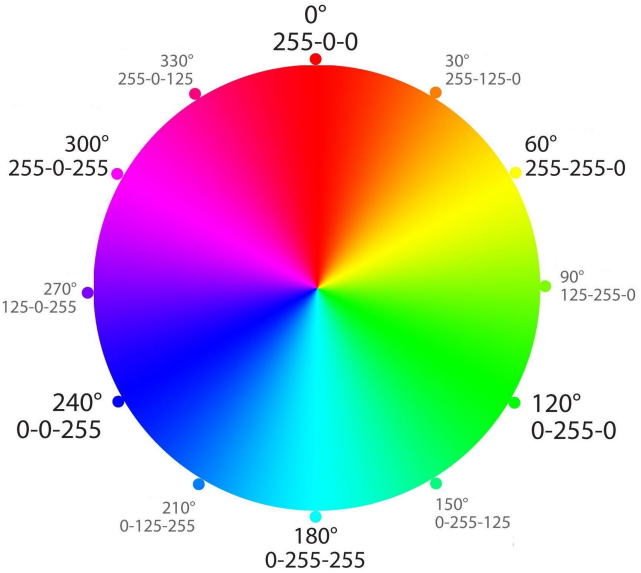
Conexiones del diodo:

1. R: a la salida 3~ de la placa con una resistencia de 220Ω.
2. (-): al raíl conectado a GND.
3. G: a la salida 5~ de la placa con una resistencia de 220Ω.
4. B: a la salida 6~ de la placa con una resistencia de 100Ω.

Conexiones del potenciómetro:

1. Raíl de alimentación (5V)
2. A la entrada analógica A0 de la placa
3. Raíl conectado a GND

Programa



El código puede resultar un poco complicado de entender, especialmente la parte del `void loop()`, pero sencillamente estamos “mapeando” es decir haciendo una equivalencia entre los valores entre 0 y 1023 que nos da el potenciómetro y el ángulo de un círculo, de 0 a 360º, esto nos permite luego separar todo en sectores de 60º para poder hacer una transición de colores equivalente a la de la imagen. Es muy recomendable, como se menciona en el propio código, aumentar el tiempo de delay (`delay()` en la última línea) en el código para poder abrir el Serial Monitor o Monitor Serie (desde el menú de Herramientas en la IDE) y poder observar los valores de la posición y código RGB de los colores junto con el ángulo equivalente.

```
#define Red 3
#define Green 5
#define Blue 6
#define Potenc A0
double valorPotenc;
double posPotenc;
double valorRed;
double valorGreen;
double valorBlue;
int posSextnt;

void setup() {
  pinMode(Red, OUTPUT);
  pinMode(Green, OUTPUT);
  pinMode(Blue, OUTPUT);
  pinMode(Potenc, INPUT);
  Serial.begin(9600);
}

void loop() {
  // leemos el valor del potenciómetro:
  valorPotenc = analogRead(Potenc);
  // mapeamos el valor que hemos obtenido (de 0 a 1023) a coordenadas
circulares (de 0 a 360 grados):
  posPotenc = 360 * (valorPotenc / 1023);
  // cambiamos el valor de la salida en función de la posición separando
en 6 casos:
  posSextnt = posPotenc / 60;
  switch (posSextnt) {
    case 0:
      valorRed = 255;
      valorGreen = 255 * posPotenc / 60; //aumenta con la posición
      valorBlue = 0;
      break;
    case 1:
      valorRed = -255 * (posPotenc - 60) / 60 + 255; //disminuye con la
posición
      valorGreen = 255;
      valorBlue = 0;
      break;
    case 2:
      valorRed = 0;
      valorGreen = 255;
      valorBlue = 255 * (posPotenc - 120) / 60;
      break;
    case 3:
```

```
      valorRed = 0;
      valorGreen = -255 * (posPotenc - 180) / 60 + 255;
      valorBlue = 255;
      break;
    case 4:
      valorRed = 255 * (posPotenc - 240) / 60;
      valorGreen = 0;
      valorBlue = 255;
      break;
    case 5:
      valorRed = 255;
      valorGreen = 0;
      valorBlue = -255 * (posPotenc - 300) / 60 + 255;
      break;
    default:
      valorRed = 255; //DDD igual falta offset
      valorGreen = 0;
      valorBlue = 0;
      break;
  }
  analogWrite(Red, valorRed);
  analogWrite(Green, valorGreen);
  analogWrite(Blue, valorBlue);

  //sacamos los valores por el Serial Monitor: (para poder verlos bien
es interesante
  // aumentar el tiempo de delay al final del programa):
  Serial.print("sensor = ");
  Serial.print(valorPotenc);
  Serial.print("\t posicion = ");
  Serial.print(posPotenc);
  Serial.print("\t sextante = ");
  Serial.print(posSextnt);
  Serial.print("\t color RGB = ");
  Serial.print(valorRed);
  Serial.print("-");
  Serial.print(valorGreen);
  Serial.print("-");
  Serial.println(valorBlue);
  // esperamos 2 milisegundos antes del próximo bucle:
  delay(2);
}
```

Más ejemplos e ideas

https://docs.arduino.cc/built-in-examples/

“Arduino Cookbook” by Michael Margolis.

“Programming Arduino”, “30 Arduino Projects for the Evil Genius” by Simon Monk.