Morse Code LED

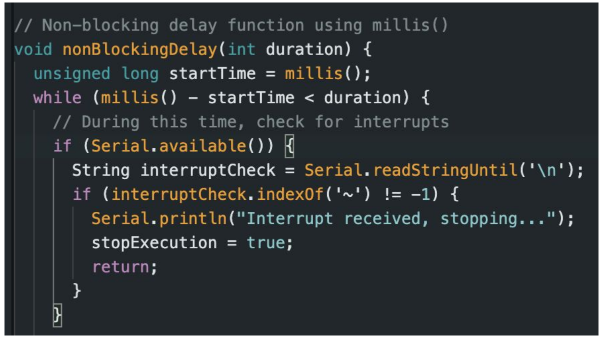
Julian Torres

Requirements

1. Take string input from user
    a. No special characters permitted
2. Display string in morse code using Arduino with LED
3. Replay morse code on LED until a specified input is received (‘~’)
    a. I chose ~ since its a simple way to get platform agnosticism compared to something like ctrl-z which may differ in keys pressed from Mac to Windows, etc. A simple compromise to make that saves potential headaches

Design

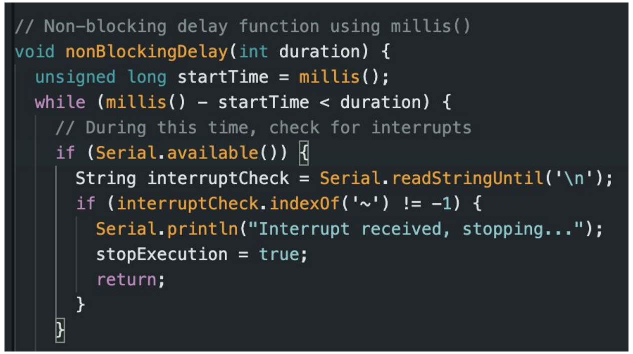
- Taking input from user
  - Using built in static method Serial.readStringUntil('\n')
  - Error handling by explicitly checking for special characters from user and exiting upon invalid input
- Custom non-blocking delay function to implement software interrupt
  - Not quite as low level as an instruction set level interrupt, but suffices for this project, since it only must simply appear to the user as being interrupted
  - In other situations, interrupts may need to be made on a lower level (e.g. hardware) to meet requirements
  - Rather than using built in delay() function, this custom one actively checks serial port while delaying



```
// Non-blocking delay function using millis()
void nonBlockingDelay(int duration) {
  unsigned long startTime = millis();
  while (millis() - startTime < duration) {
    // During this time, check for interrupts
    if (Serial.available()) {
      String interruptCheck = Serial.readStringUntil('\n');
      if (interruptCheck.indexOf('~') != -1) {
        Serial.println("Interrupt received, stopping...");
        stopExecution = true;
        return;
      }
    }
```

Design

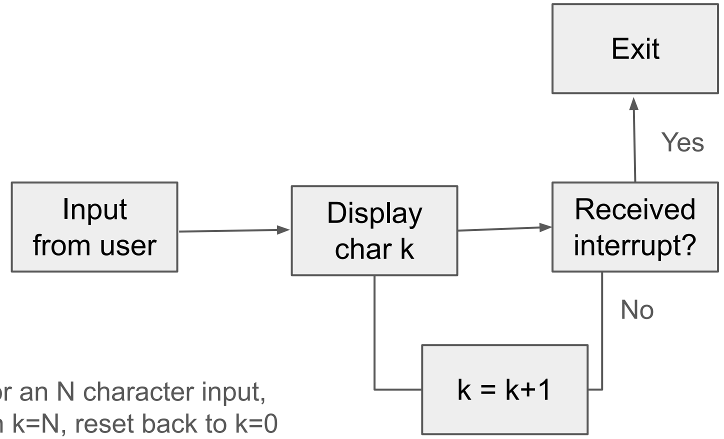
- Custom non-blocking delay function to implement software interrupt
  - Not quite as low level as an instruction set level interrupt, but suffices for this project, since it only must simply appear to the user as being interrupted
  - In other situations, interrupts may need to be made on a lower level (eq. hardware) to meet requirements
  - Rather than using built in delay() function, this custom one actively checks serial port while delaying



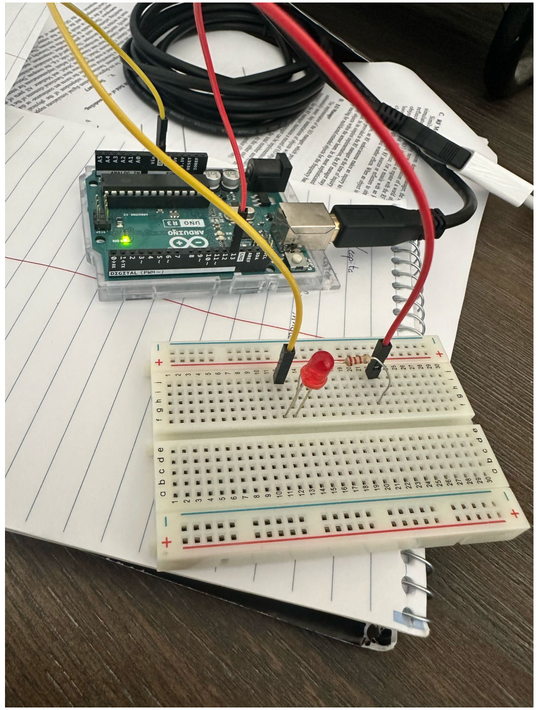
```
// Non-blocking delay function using millis()
void nonBlockingDelay(int duration) {
  unsigned long startTime = millis();
  while (millis() - startTime < duration) {
    // During this time, check for interrupts
    if (Serial.available()) {
      String interruptCheck = Serial.readStringUntil('\n');
      if (interruptCheck.indexOf('~') != -1) {
        Serial.println("Interrupt received, stopping...");
        stopExecution = true;
        return;
      }
    }
```

Implementation

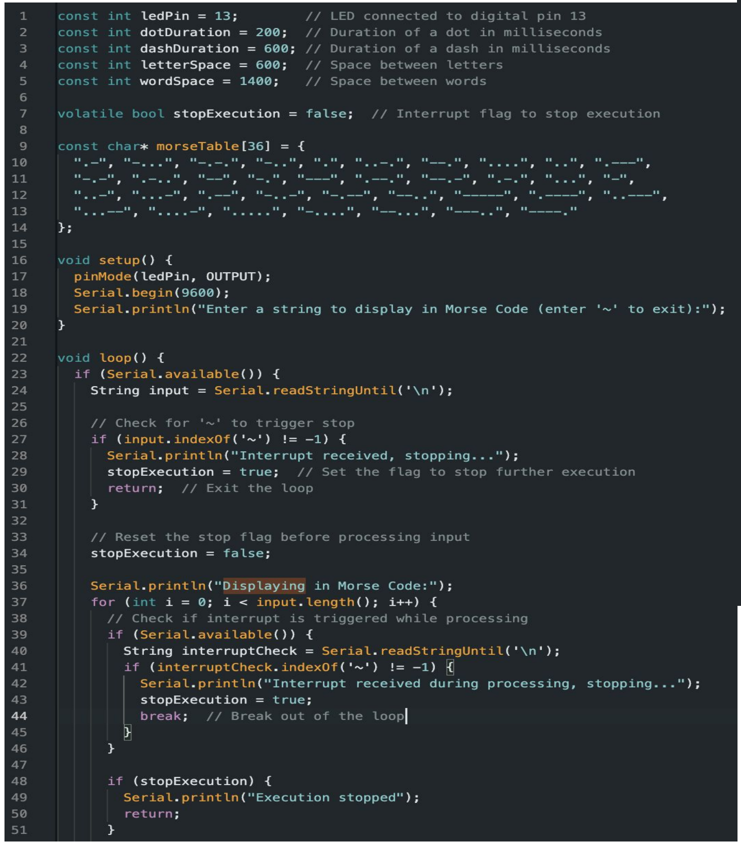
- Simple Arduino circuit with only an LED and load resistor
- Logic:



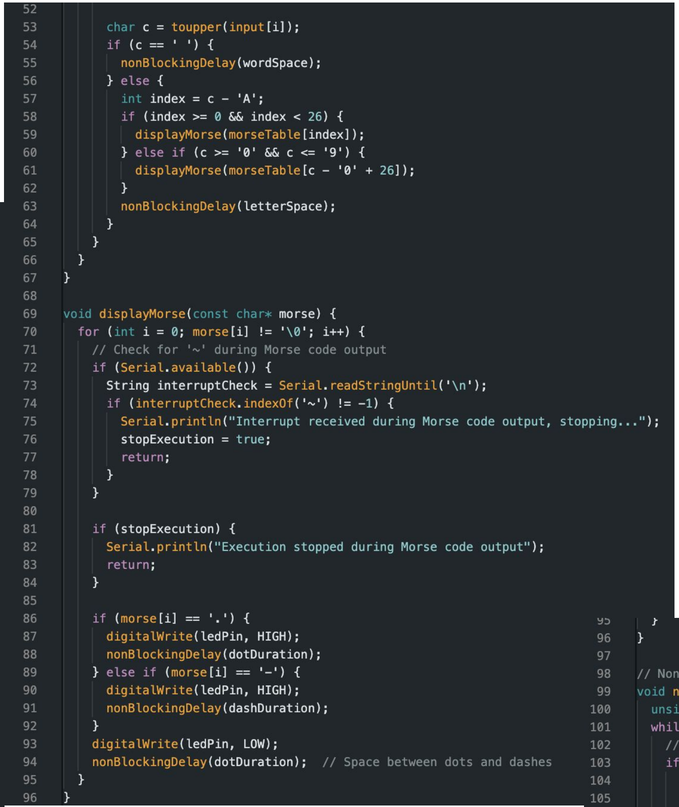
***For an N character input, when k=N, reset back to k=0



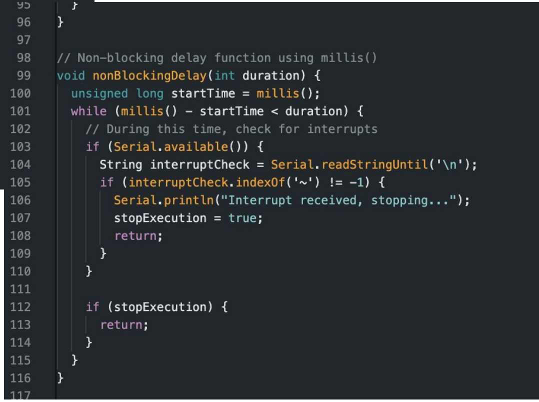
Full Code



```cpp
const int ledPin = 13;         // LED connected to digital pin 13
const int dotDuration = 200;  // Duration of a dot in milliseconds
const int dashDuration = 600; // Duration of a dash in milliseconds
const int letterSpace = 600;  // Space between letters
const int wordSpace = 1400;   // Space between words

volatile bool stopExecution = false;  // Interrupt flag to stop execution

const char* morseTable[36] = {
  ".-", "-...", "-.-.", "-..", ".", "..-.", "--.", "....", "..", ".---",
  "-.-", ".-..", "--", "-.", "---", ".--.", "--.-", ".-.", "...", "-",
  "..-", "...-", ".--", "-..-", "-.--", "--..", "-----", ".----", "..---",
  "...--", "....-", ".....", "-....", "--...", "---..", "----."
};

void setup() {
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
  Serial.println("Enter a string to display in Morse Code (enter '~' to exit):");
}

void loop() {
  if (Serial.available()) {
    String input = Serial.readStringUntil('\n');

    // Check for '~' to trigger stop
    if (input.indexOf('~') != -1) {
      Serial.println("Interrupt received, stopping...");
      stopExecution = true;  // Set the flag to stop further execution
      return;  // Exit the loop
    }

    // Reset the stop flag before processing input
    stopExecution = false;

    Serial.println("Displaying in Morse Code:");
    for (int i = 0; i < input.length(); i++) {
      // Check if interrupt is triggered while processing
      if (Serial.available()) {
        String interruptCheck = Serial.readStringUntil('\n');
        if (interruptCheck.indexOf('~') != -1) {
          Serial.println("Interrupt received during processing, stopping...");
          stopExecution = true;
          break;  // Break out of the loop
        }
      }

      if (stopExecution) {
        Serial.println("Execution stopped");
        return;
      }
```



```cpp

      char c = toupper(input[i]);
      if (c == ' ') {
        nonBlockingDelay(wordSpace);
      } else {
        int index = c - 'A';
        if (index >= 0 && index < 26) {
          displayMorse(morseTable[index]);
        } else if (c >= '0' && c <= '9') {
          displayMorse(morseTable[c - '0' + 26]);
        }
        nonBlockingDelay(letterSpace);
      }
    }
  }
}

void displayMorse(const char* morse) {
  for (int i = 0; morse[i] != '\0'; i++) {
    // Check for '~' during Morse code output
    if (Serial.available()) {
      String interruptCheck = Serial.readStringUntil('\n');
      if (interruptCheck.indexOf('~') != -1) {
        Serial.println("Interrupt received during Morse code output, stopping...");
        stopExecution = true;
        return;
      }
    }

    if (stopExecution) {
      Serial.println("Execution stopped during Morse code output");
      return;
    }

    if (morse[i] == '.') {
      digitalWrite(ledPin, HIGH);
      nonBlockingDelay(dotDuration);
    } else if (morse[i] == '-') {
      digitalWrite(ledPin, HIGH);
      nonBlockingDelay(dashDuration);
    }
    digitalWrite(ledPin, LOW);
    nonBlockingDelay(dotDuration);  // Space between dots and dashes
  }
}
```



```cpp
  }
}

// Non-blocking delay function using millis()
void nonBlockingDelay(int duration) {
  unsigned long startTime = millis();
  while (millis() - startTime < duration) {
    // During this time, check for interrupts
    if (Serial.available()) {
      String interruptCheck = Serial.readStringUntil('\n');
      if (interruptCheck.indexOf('~') != -1) {
        Serial.println("Interrupt received, stopping...");
        stopExecution = true;
        return;
      }
    }

    if (stopExecution) {
      return;
    }
  }
}
```