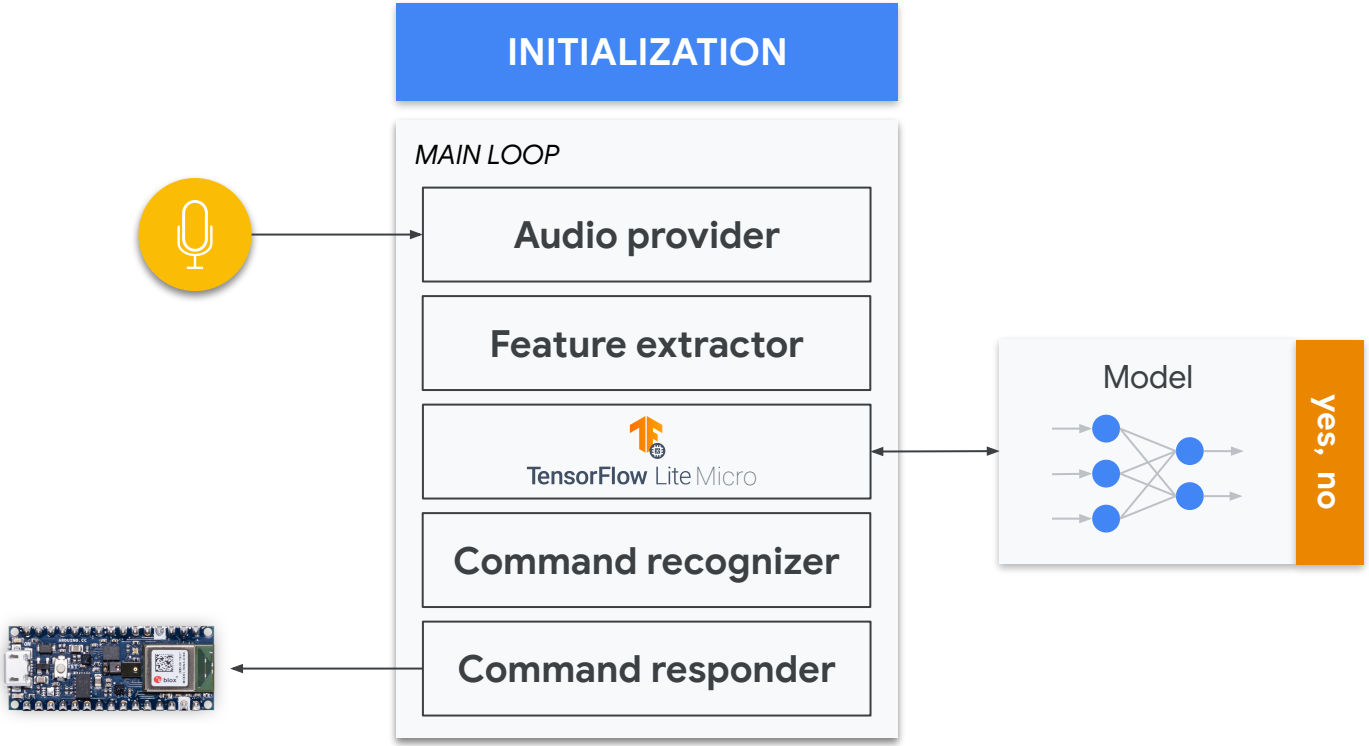
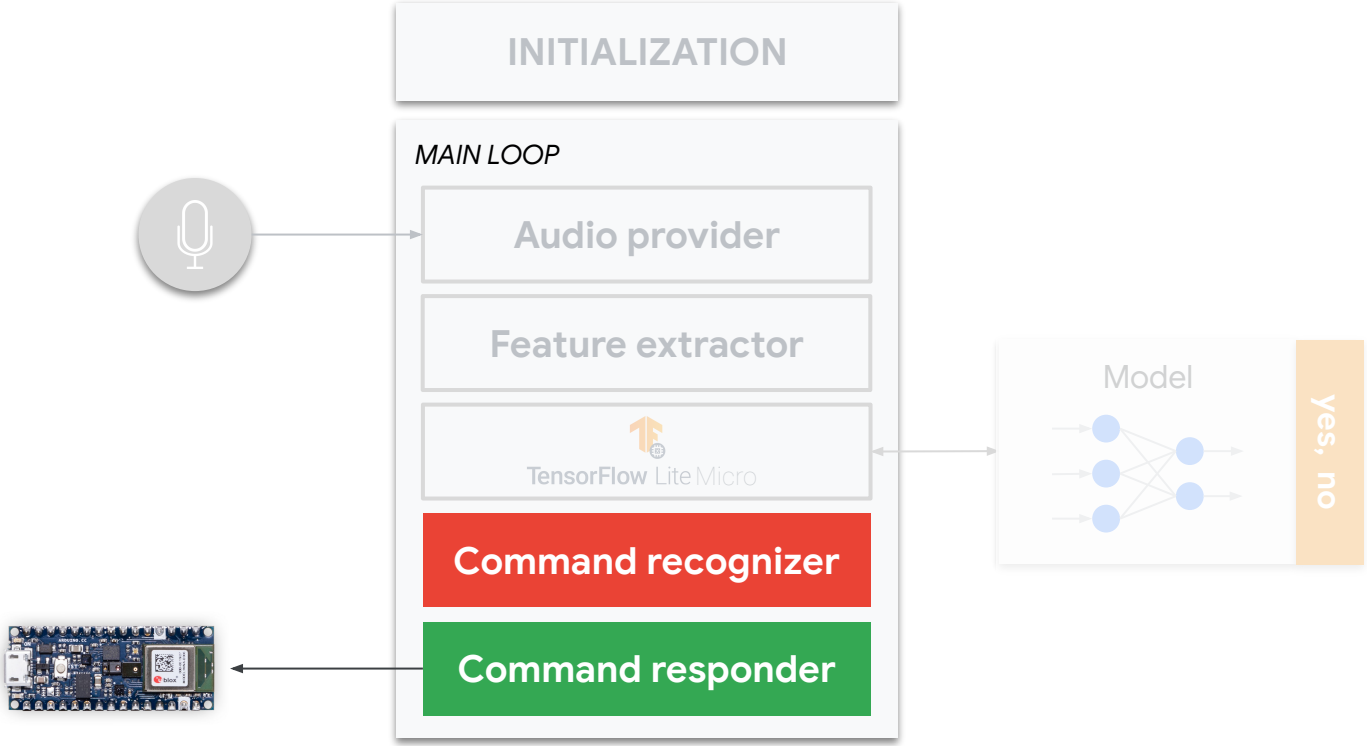
KWS Post-processing


INITIALIZATION
MAIN LOOP
Audio provider
Feature extractor
TensorFlow Lite Micro
Command recognizer
Command responder
Model
yes, no


INITIALIZATION
MAIN LOOP
Audio provider
Feature extractor
TensorFlow Lite Micro
Command recognizer
Command responder
Model
yes, no

user says,

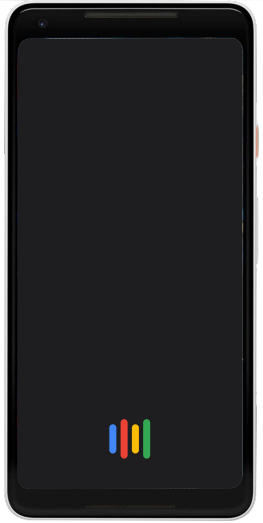
“Upward!”

Command recognizer

Command responder

Keyword

“Up”



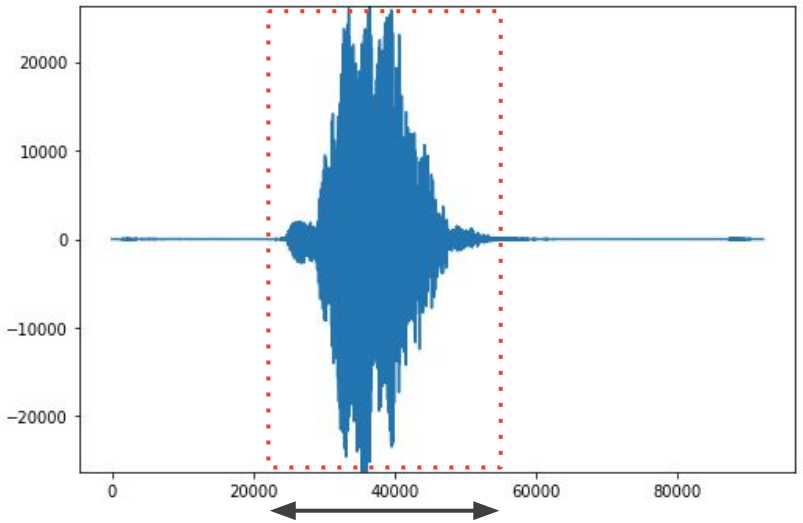
?

Command recognizer

Command responder

user says,

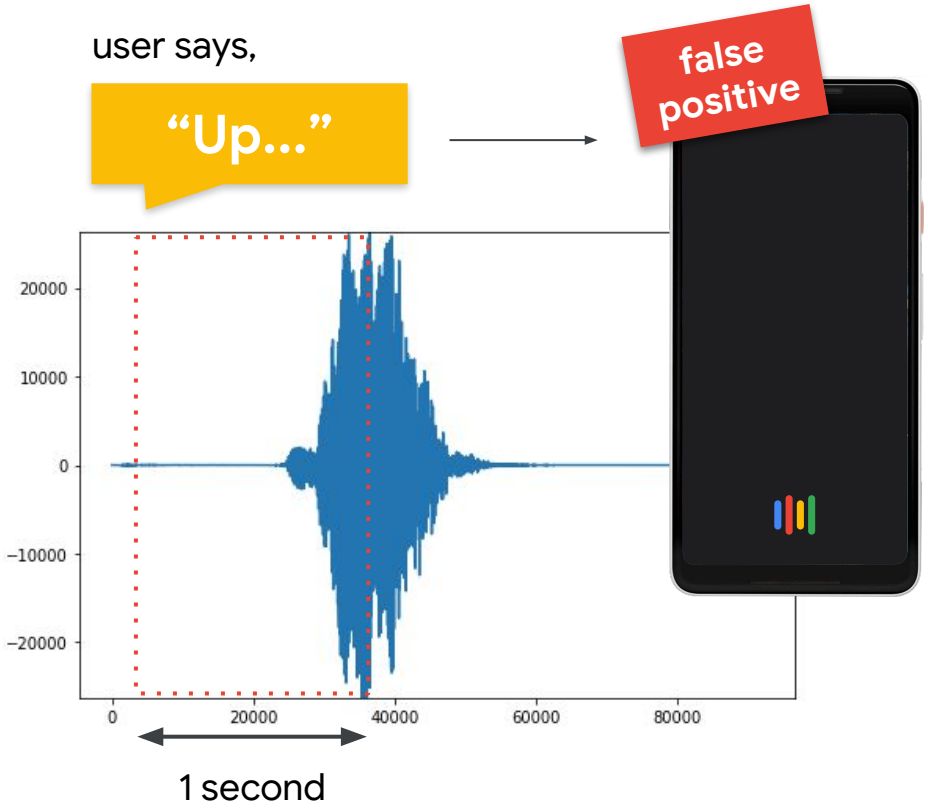
"Upward!"



1 second

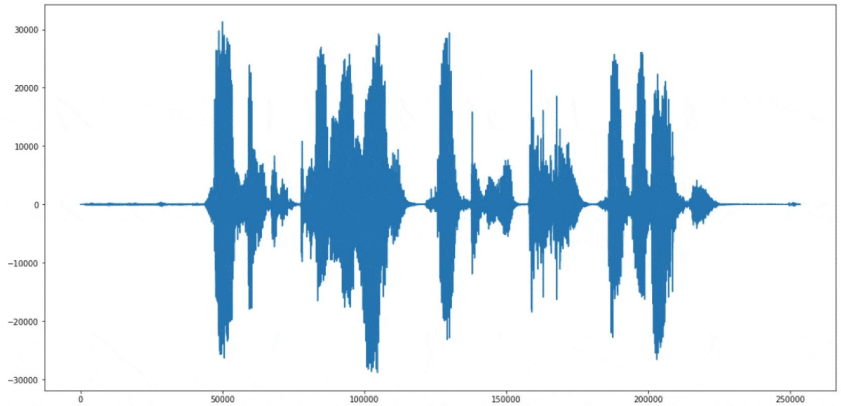
Command recognizer

Command responder



Command recognizer

Command responder

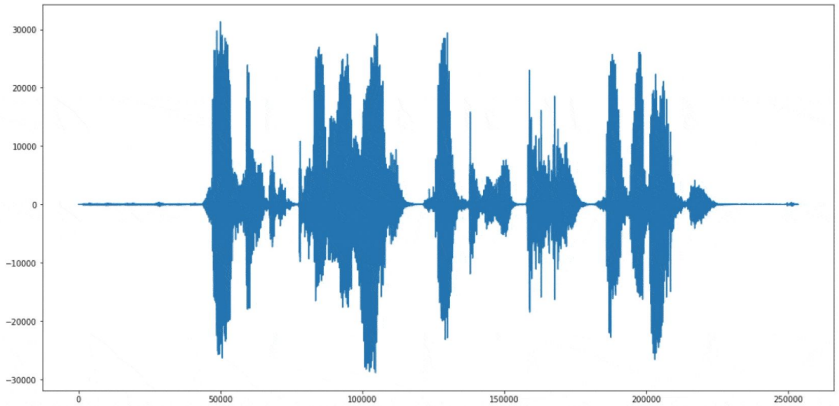


For every *new* window:

1. **Store** new inference
2. **Calculate** new score for all of the keywords based on new inference
3. **Output** new average scores

Command recognizer

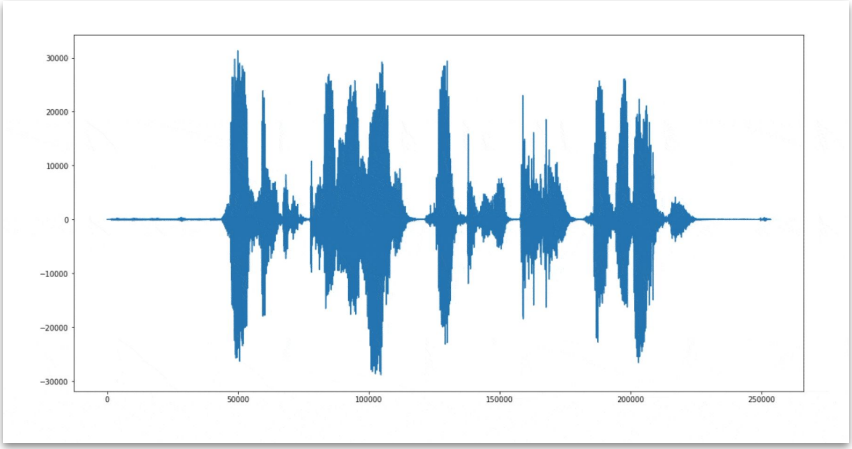
Command responder



Timestep	“Yes”	“No”	“Unknown”
1	55%	35%	20%
2	65%	25%	10%
3	76%	12%	12%
4	88%	7%	5%
5	99%	0.5%	0.5%

Command recognizer

Command responder




```
// Determine whether a command was recognized based
// on the output of inference

const char* found_command = nullptr;
uint8_t score = 0;
bool is_new_command = false;

TfLiteStatus process_status = recognizer->ProcessLatestResults(
    output, current_time, &found_command, &score, &is_new_command);

if (process_status != kTfLiteOk) {
      TF_LITE_REPORT_ERROR(error_reporter,
      "RecognizeCommands::ProcessLatestResults() failed");
      return;
}
```

Command recognizer

Command responder

```
// Do something based on the recognized command.
// The default implementation just prints to
// the error console, but you should replace with
// your own function for a real application.

RespondToCommand(error_reporter, current_time,
                 found_command, score, is_new_command);
```

Command recognizer

Command responder

```
if (is_new_command) {
  TF_LITE_REPORT_ERROR(error_reporter, "Heard %s (%d) @%dms",
found_command,
                       score, current_time);
  // If we hear a command, light up the appropriate LED
  if (found_command[0] == 'y') {
    last_command_time = current_time;
    digitalWrite(LEDG, LOW);  // Green for yes
  }

  if (found_command[0] == 'n') {
    last_command_time = current_time;
    digitalWrite(LEDR, LOW);  // Red for no
  }

  if (found_command[0] == 'u') {
    last_command_time = current_time;
    digitalWrite(LEDB, LOW);  // Blue for unknown
  }
}
```

Command recognizer

Command responder