

# IESTI01 - TinyML

Person Detection (VWW)  
Application

Prof. Marcelo Rovai

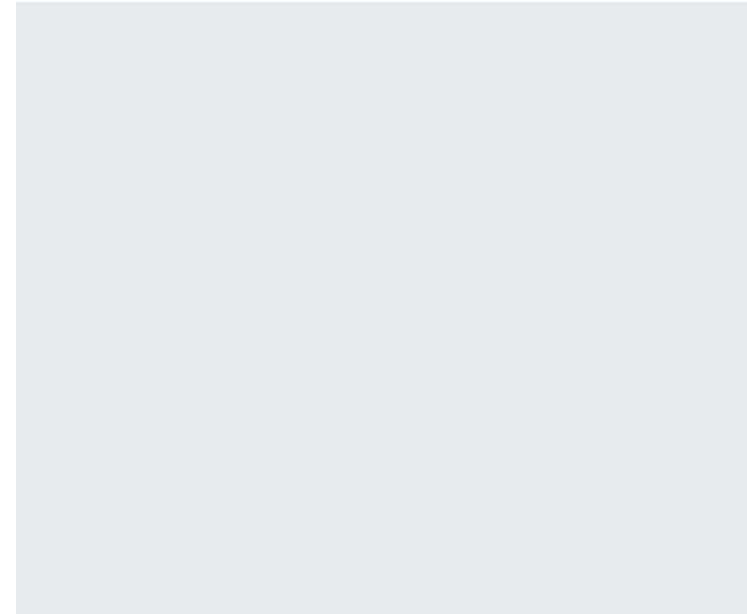
July 21<sup>st</sup>, 2021



# Person Detection: Application Architecture



✓

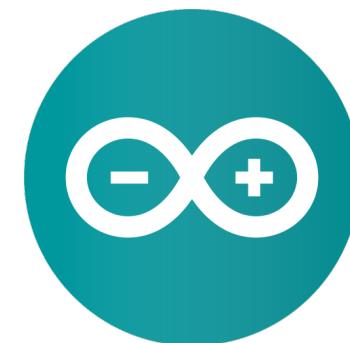


X

# Person Detection using Transfer Learning Model

## Code Walkthrough!

person\_detection.ino





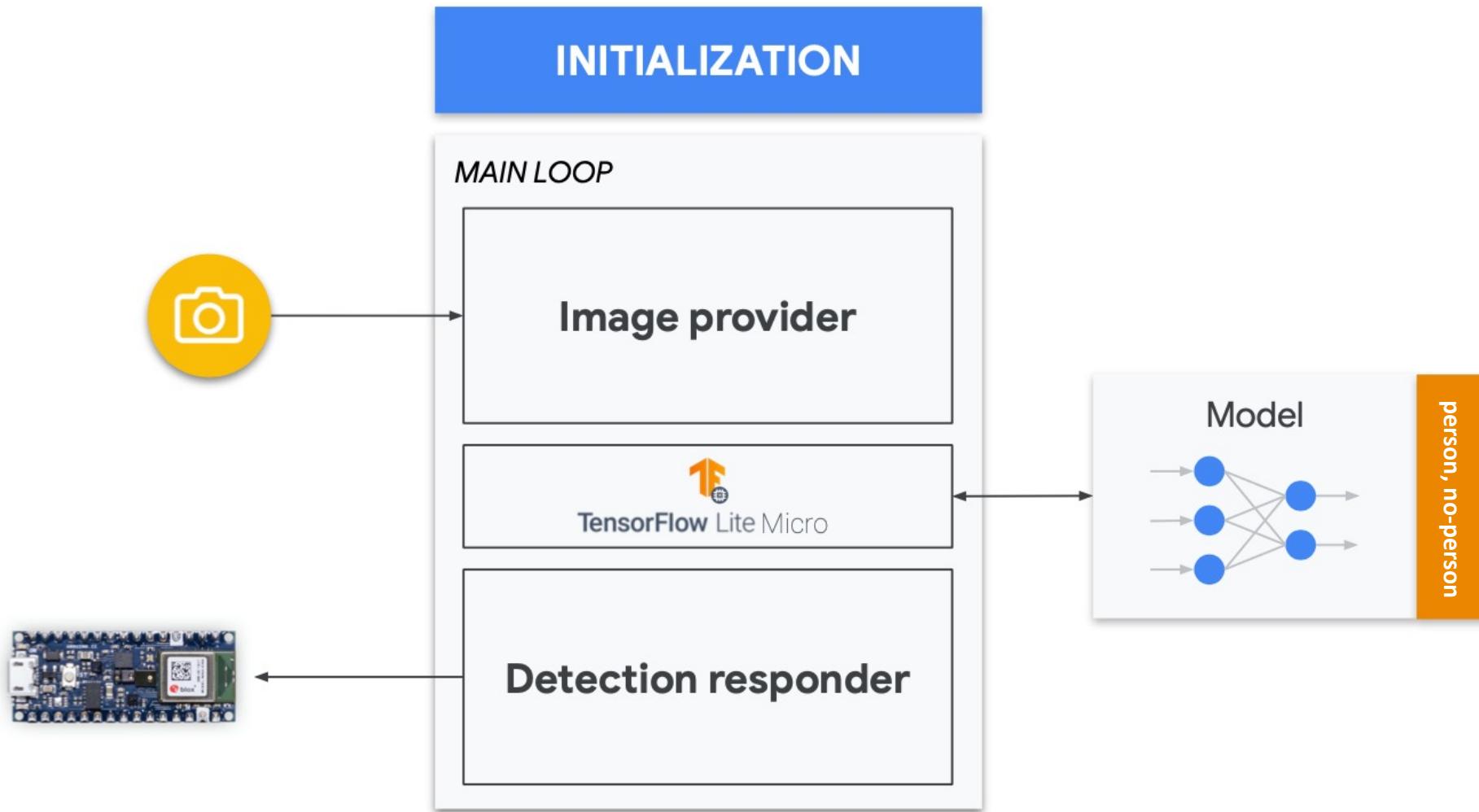
[TensorFlow Lite Micro - Paper](#)



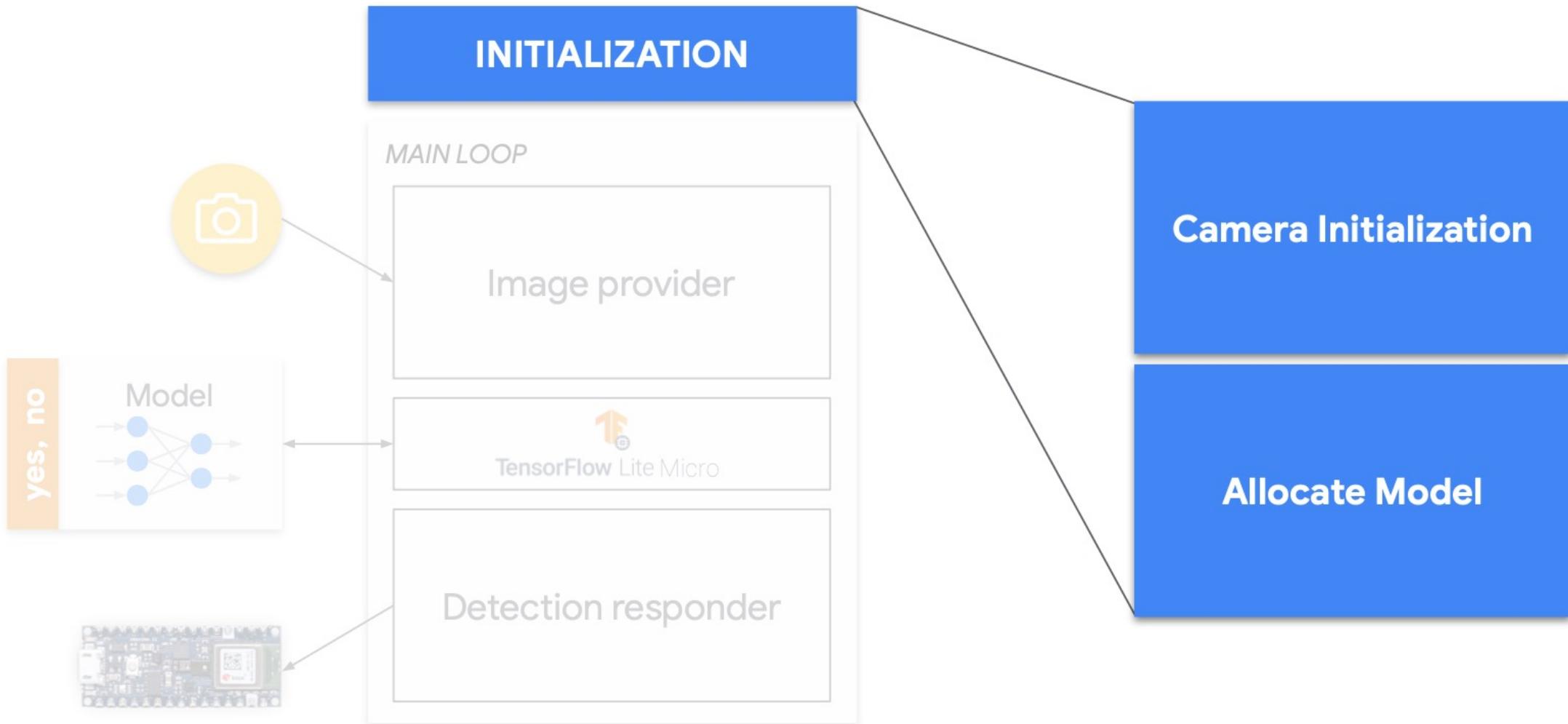
[MLSys 2021: TensorFlow Lite Micro TFLM](#)



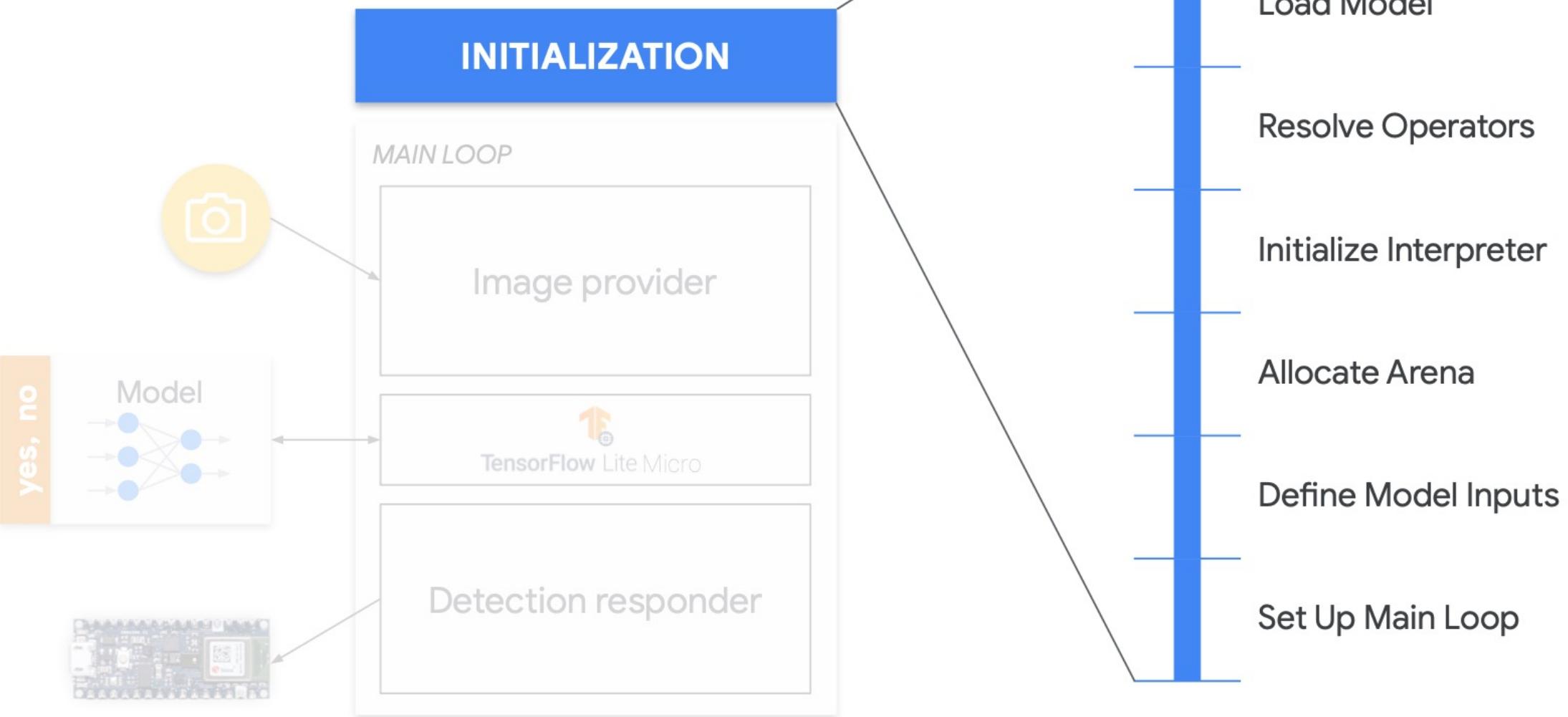
# Person Detection Components



# Initialization



# Initialization



```
person_detection | Arduino 1.8.15

person_detection arduino_detection_responder.cpp arduino_image_provider.cpp arduino_main.cpp detection_responder.h image_provider.h

15
16 #include <TensorFlowLite.h>
17
18 #include "main_functions.h"
19
20 #include "detection_responder.h"
21 #include "image_provider.h"
22 #include "model_settings.h"
23 #include "person_detect_model_data.h"
24 #include "tensorflow/lite/micro/micro_error_reporter.h"
25 #include "tensorflow/lite/micro/micro_interpreter.h"
26 #include "tensorflow/lite/micro/micro_mutable_op_resolver.h"
27 #include "tensorflow/lite/schema/schema_generated.h"
28 #include "tensorflow/lite/version.h"
29
30 // Globals, used for compatibility with Arduino-style sketches.
31 namespace {
32     tflite::ErrorReporter* error_reporter = nullptr;
33     const tflite::Model* model = nullptr;
34     tflite::MicroInterpreter* interpreter = nullptr;
35     TfLiteTensor* input = nullptr;
36
37     // In order to use optimized tensorflow lite kernels, a signed int8_t quantized
38     // model is preferred over the legacy unsigned model format. This means that
39     // throughout this project, input images must be converted from unsigned to
40     // signed format. The easiest and quickest way to convert from unsigned to
41     // signed 8-bit integers is to subtract 128 from the unsigned value to get a
42     // signed value.
43
44     // An area of memory to use for input, output, and intermediate arrays.
45     constexpr int kTensorArenaSize = 136 * 1024;
46     static uint8_t tensor_arena[kTensorArenaSize];
47 } // namespace
48
```

## Declare Variables

Load Model

Resolve Operators

Initialize Interpreter

Allocate Arena

Define Model Inputs

Set Up Main Loop

```
person_detection | Arduino 1.8.15
person_detection arduino_detection_responder.cpp arduino_image_provider.cpp arduino_main.cpp detection_responder.h image_provider.h
50 void setup() {
51     // Set up logging. Google style is to avoid globals or statics because of
52     // lifetime uncertainty, but since this has a trivial destructor it's okay.
53     // NOLINTNEXTLINE(runtime-global-variables)
54     static tflite::MicroErrorReporter micro_error_reporter;
55     error_reporter = &micro_error_reporter;
56
57     // Map the model into a usable data structure. This doesn't involve any
58     // copying or parsing, it's a very lightweight operation.
59     model = tflite::GetModel(g_person_detect_model_data);
60     if (model->version() != TFLITE_SCHEMA_VERSION) {
61         TF_LITE_REPORT_ERROR(error_reporter,
62             "Model provided is schema version %d not equal "
63             "to supported version %d.",
64             model->version(), TFLITE_SCHEMA_VERSION);
65         return;
66     }
67
68     // Pull in only the operation implementations we need.
69     // This relies on a complete list of all the ops needed by this graph.
70     // An easier approach is to just use the AllOpsResolver, but this will
71     // incur some penalty in code space for op implementations that are not
72     // needed by this graph.
73     //
74     // tflite::AllOpsResolver resolver;
75     // NOLINTNEXTLINE(runtime-global-variables)
76     static tflite::MicroMutableOpResolver<5> micro_op_resolver;
77     micro_op_resolver.AddAveragePool2D();
78     micro_op_resolver.AddConv2D();
79     micro_op_resolver.AddDepthwiseConv2D();
80     micro_op_resolver.AddReshape();
81     micro_op_resolver.AddSoftmax();
82 }
```

Declare Variables

Load Model

Resolve Operators

Initialize Interpreter

Allocate Arena

Define Model Inputs

Set Up Main Loop

```
23
24 // Keep model aligned to 8 bytes to guarantee aligned 64-bit accesses.
25 const unsigned char g_person_detect_model_data[] = {
26     0x1c, 0x00, 0x00, 0x00, 0x54, 0x46, 0x4c, 0x33, 0x00, 0x00, 0x00, 0x00,
27     0x00, 0xe, 0x00, 0x18, 0x00, 0x4, 0x0, 0x8, 0x00, 0xc, 0x00, 0x10, 0x00,
28     0x14, 0x00, 0xe, 0x00, 0x00, 0x03, 0x00, 0x00, 0x84, 0x95, 0x4,
29     0x00, 0xec, 0xb, 0x3, 0x00, 0xd4, 0xb, 0x3, 0x00, 0x4, 0x0, 0x0, 0x00,
30     0x5a, 0x00, 0x00, 0x00, 0xc4, 0x5b, 0x03, 0x00, 0xac, 0x5b, 0x03, 0x00, 0x94,
31     0x5b, 0x3, 0x00, 0x84, 0x59, 0x03, 0x00, 0x74, 0x55, 0x03, 0x00, 0x64, 0x55,
32     0x2, 0x00, 0x54, 0x51, 0x02, 0x00, 0x44, 0x48, 0x02, 0x00, 0x34, 0x44, 0x02,
33     0x00, 0x24, 0x42, 0x02, 0x00, 0x94, 0x3d, 0x02, 0x00, 0x84, 0x3b, 0x02, 0x00,
34     0x74, 0xfb, 0x1, 0x00, 0xe4, 0xf6, 0x1, 0x00, 0xd4, 0xb6, 0x1, 0x00, 0xc4,
35     0xb4, 0x1, 0x00, 0xb4, 0x74, 0x1, 0x00, 0xa4, 0x72, 0x1, 0x00, 0x94, 0x70,
36     0x1, 0x00, 0x84, 0x6e, 0x1, 0x00, 0x74, 0x2e, 0x1, 0x00, 0x64, 0xee, 0x00,
37     0x00, 0x54, 0xec, 0x00, 0x00, 0xc4, 0xe7, 0x00, 0x00, 0xb4, 0xe5, 0x00, 0x00,
38     0xa4, 0xc5, 0x00, 0x00, 0x94, 0xc4, 0x00, 0x00, 0x44, 0xc2, 0x00, 0x00, 0x34,
39     0xb2, 0x00, 0x00, 0x24, 0xb1, 0x00, 0x00, 0x14, 0xa9, 0x00, 0x00, 0x84, 0xa8,
40     0x00, 0x00, 0x54, 0xa7, 0x00, 0x00, 0x44, 0xa3, 0x00, 0x00, 0xb4, 0xa2, 0x00,
41     0x00, 0x84, 0xa1, 0x00, 0x00, 0x34, 0xa1, 0x00, 0x00, 0x2c, 0xa1, 0x00, 0x00,
42     0x24, 0xa1, 0x00, 0x00, 0x1c, 0xa1, 0x00, 0x00, 0x14, 0xa1, 0x00, 0x00, 0x0c,
43     0xa1, 0x00, 0x04, 0xa1, 0x00, 0x00, 0xfc, 0xa0, 0x00, 0x00, 0xf4, 0xa0,
44     0x00, 0x00, 0xec, 0xa0, 0x00, 0x00, 0xe4, 0xa0, 0x00, 0x00, 0xdc, 0xa0, 0x00,
45     0x00, 0x8c, 0xa0, 0x00, 0x00, 0x84, 0xa0, 0x00, 0x00, 0x7c, 0xa0, 0x00, 0x00,
46     0x74, 0xa0, 0x00, 0x00, 0x6c, 0xa0, 0x00, 0x00, 0x64, 0xa0, 0x00, 0x00, 0x5c,
47     0xa0, 0x00, 0x4c, 0x9e, 0x00, 0x00, 0x1c, 0x9e, 0x00, 0x00, 0x14, 0x9e,
48     0x00, 0x00, 0x74, 0x9d, 0x00, 0x00, 0xe4, 0x9c, 0x00, 0x00, 0x8c, 0x9c, 0x00,
49     0x00, 0x7c, 0x9a, 0x00, 0x00, 0xec, 0x99, 0x00, 0x00, 0x5c, 0x99, 0x00, 0x00,
50     0x54, 0x99, 0x00, 0x4c, 0x99, 0x00, 0x00, 0x44, 0x99, 0x00, 0x00, 0x3c,
51     0x99, 0x00, 0x00, 0xe4, 0x98, 0x00, 0x00, 0xd4, 0x18, 0x00, 0x00, 0xc4, 0x16,
52     0x00, 0x00, 0x34, 0x12, 0x00, 0x00, 0xa4, 0xd, 0x00, 0x00, 0x9c, 0xd, 0x00,
53     0x00, 0x94, 0xd, 0x00, 0x00, 0x8c, 0xd, 0x00, 0x00, 0x7c, 0xc, 0x00, 0x00,
54     0x2c, 0xa, 0x00, 0x00, 0x1c, 0x08, 0x00, 0x00, 0x14, 0x08, 0x00, 0x00, 0x84,
55     0x03, 0x00, 0x00, 0x7c, 0x03, 0x00, 0x00, 0x4c, 0x03, 0x00, 0x00, 0x3c, 0x01,
56     0x00, 0x00, 0x2c, 0x00, 0x00, 0x00, 0x24, 0x00, 0x00, 0x1c, 0x00, 0x00,
```

Declare Variables

Load Model

Resolve Operators

Initialize Interpreter

Allocate Arena

Define Model Inputs

Set Up Main Loop

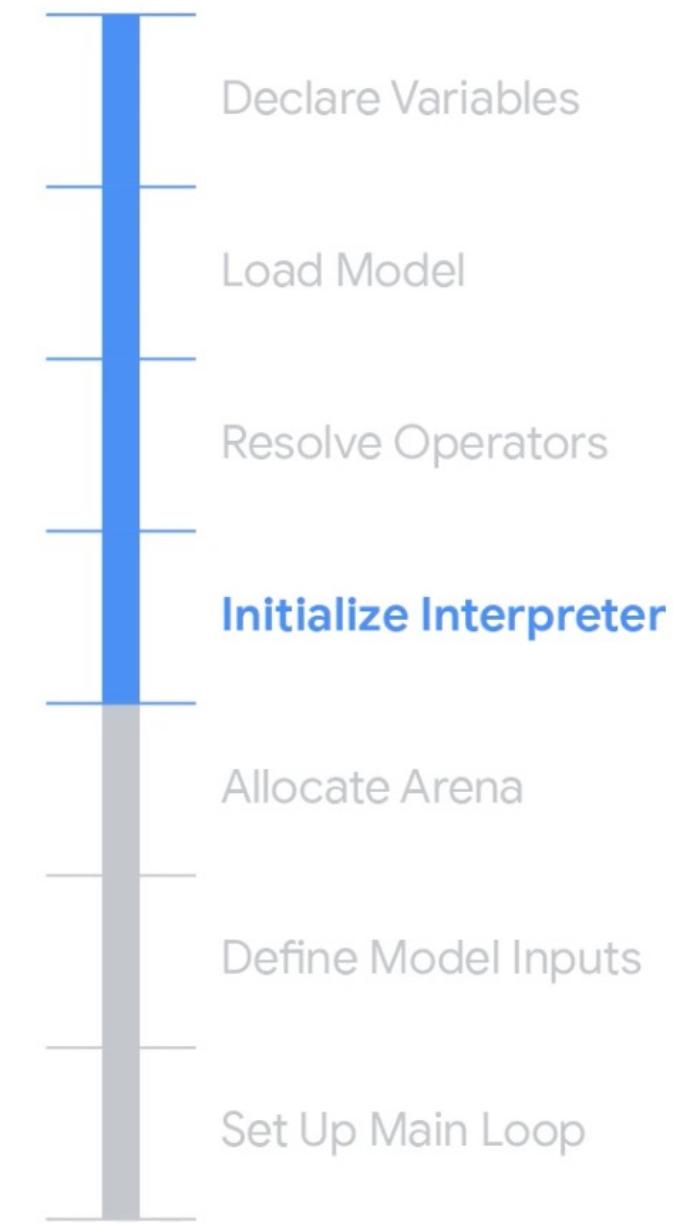
```
person_detection | Arduino 1.8.15

person_detection arduino_detection_responder.cpp arduino_image_provider.cpp arduino_main.cpp detection_responder.h image_provider.h

50 void setup() {
51     // Set up logging. Google style is to avoid globals or statics because of
52     // lifetime uncertainty, but since this has a trivial destructor it's okay.
53     // NOLINTNEXTLINE(runtime-global-variables)
54     static tflite::MicroErrorReporter micro_error_reporter;
55     error_reporter = &micro_error_reporter;
56
57     // Map the model into a usable data structure. This doesn't involve any
58     // copying or parsing, it's a very lightweight operation.
59     model = tflite::GetModel(g_person_detect_model_data);
60     if (model->version() != TFLITE_SCHEMA_VERSION) {
61         TF_LITE_REPORT_ERROR(error_reporter,
62             "Model provided is schema version %d not equal "
63             "to supported version %d.",
64             model->version(), TFLITE_SCHEMA_VERSION);
65
66         return;
67     }
68
69     // Pull in only the operation implementations we need.
70     // This relies on a complete list of all the ops needed by this graph.
71     // An easier approach is to just use the AllOpsResolver, but this will
72     // incur some penalty in code space for op implementations that are not
73     // needed by this graph.
74
75     // tflite::AllOpsResolver resolver;
76     // NOLINTNEXTLINE(runtime-global-variables)
77     static tflite::MicroMutableOpResolver<5> micro_op_resolver;
78     micro_op_resolver.AddAveragePool2D();
79     micro_op_resolver.AddConv2D();
80     micro_op_resolver.AddDepthwiseConv2D();
81     micro_op_resolver.AddReshape();
82     micro_op_resolver.AddSoftmax();
```



```
person_detection | Arduino 1.8.15
person_detection arduino_detection_responder.cpp arduino_image_provider.cpp arduino_main.cpp detection_responder.h image_provider.h
66 }
67
68 // Pull in only the operation implementations we need.
69 // This relies on a complete list of all the ops needed by this graph.
70 // An easier approach is to just use the AllOpsResolver, but this will
71 // incur some penalty in code space for op implementations that are not
72 // needed by this graph.
73 //
74 // tflite::AllOpsResolver resolver;
75 // NOLINTNEXTLINE(runtime-global-variables)
76 static tflite::MicroMutableOpResolver<5> micro_op_resolver;
77 micro_op_resolver.AddAveragePool2D();
78 micro_op_resolver.AddConv2D();
79 micro_op_resolver.AddDepthwiseConv2D();
80 micro_op_resolver.AddReshape();
81 micro_op_resolver.AddSoftmax();
82
83 // Build an interpreter to run the model with.
84 // NOLINTNEXTLINE(runtime-global-variables)
85 static tflite::MicroInterpreter static_interpreter(
86     model, micro_op_resolver, tensor_arena, kTensorArenaSize, error_reporter);
87 interpreter = &static_interpreter;
88
89 // Allocate memory from the tensor_arena for the model's tensors.
90 TfLiteStatus allocate_status = interpreter->AllocateTensors();
91 if (allocate_status != kTfLiteOk) {
92     TF_LITE_REPORT_ERROR(error_reporter, "AllocateTensors() failed");
93     return;
94 }
95
96 // Get information about the memory area to use for the model's input.
97 input = interpreter->input(0);
98 }
99 }
```



```
person_detection | Arduino 1.8.15
person_detection arduino_detection_responder.cpp arduino_image_provider.cpp arduino_main.cpp detection_responder.h image_provider.h
66 }
67
68 // Pull in only the operation implementations we need.
69 // This relies on a complete list of all the ops needed by this graph.
70 // An easier approach is to just use the AllOpsResolver, but this will
71 // incur some penalty in code space for op implementations that are not
72 // needed by this graph.
73 //
74 // tflite::AllOpsResolver resolver;
75 // NOLINTNEXTLINE(runtime-global-variables)
76 static tflite::MicroMutableOpResolver<5> micro_op_resolver;
77 micro_op_resolver.AddAveragePool2D();
78 micro_op_resolver.AddConv2D();
79 micro_op_resolver.AddDepthwiseConv2D();
80 micro_op_resolver.AddReshape();
81 micro_op_resolver.AddSoftmax();
82
83 // Build an interpreter to run the model with.
84 // NOLINTNEXTLINE(runtime-global-variables)
85 static tflite::MicroInterpreter static_interpreter(
86     model, micro_op_resolver, tensor_arena, kTensorArenaSize, error_reporter);
87 interpreter = &static_interpreter;
88
89 // Allocate memory from the tensor_arena for the model's tensors.
90 TfLiteStatus allocate_status = interpreter->AllocateTensors();
91 if (allocate_status != kTfLiteOk) {
92     TF_LITE_REPORT_ERROR(error_reporter, "AllocateTensors() failed");
93     return;
94 }
95
96 // Get information about the memory area to use for the model's input.
97 input = interpreter->input(0);
98 }
99 }
```

Declare Variables

Load Model

Resolve Operators

Initialize Interpreter

Allocate Arena

Define Model Inputs

Set Up Main Loop

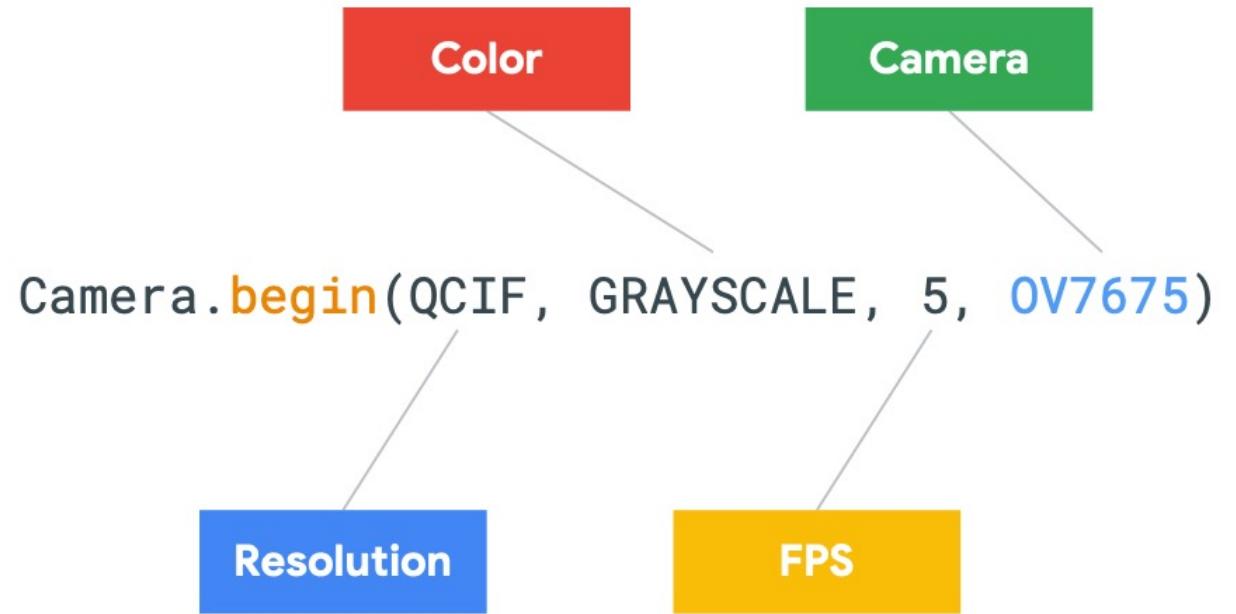
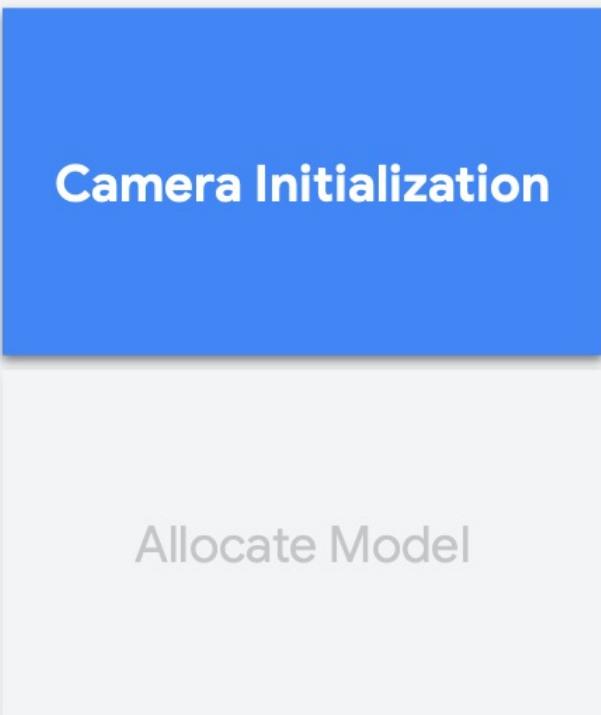
# Initialization

Camera Initialization

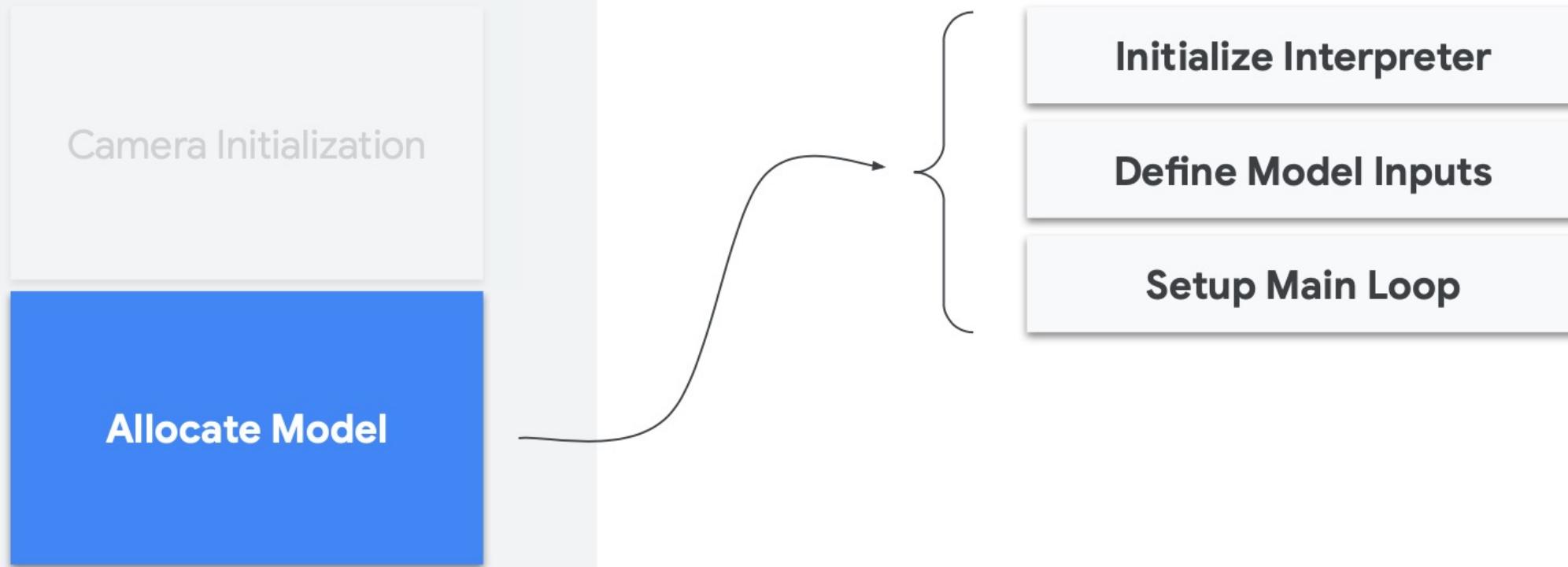
Allocate Model

```
// Initialize camera if necessary
if (!g_is_camera_initialized) {
    if (!Camera.begin(QCIF, GRayscale, 5, OV7675)) {
        TF_LITE_REPORT_ERROR(error_reporter, "Failed to
                                         initialize
                                         camera!");
        return kTfLiteError;
    }
    g_is_camera_initialized = true;
}
```

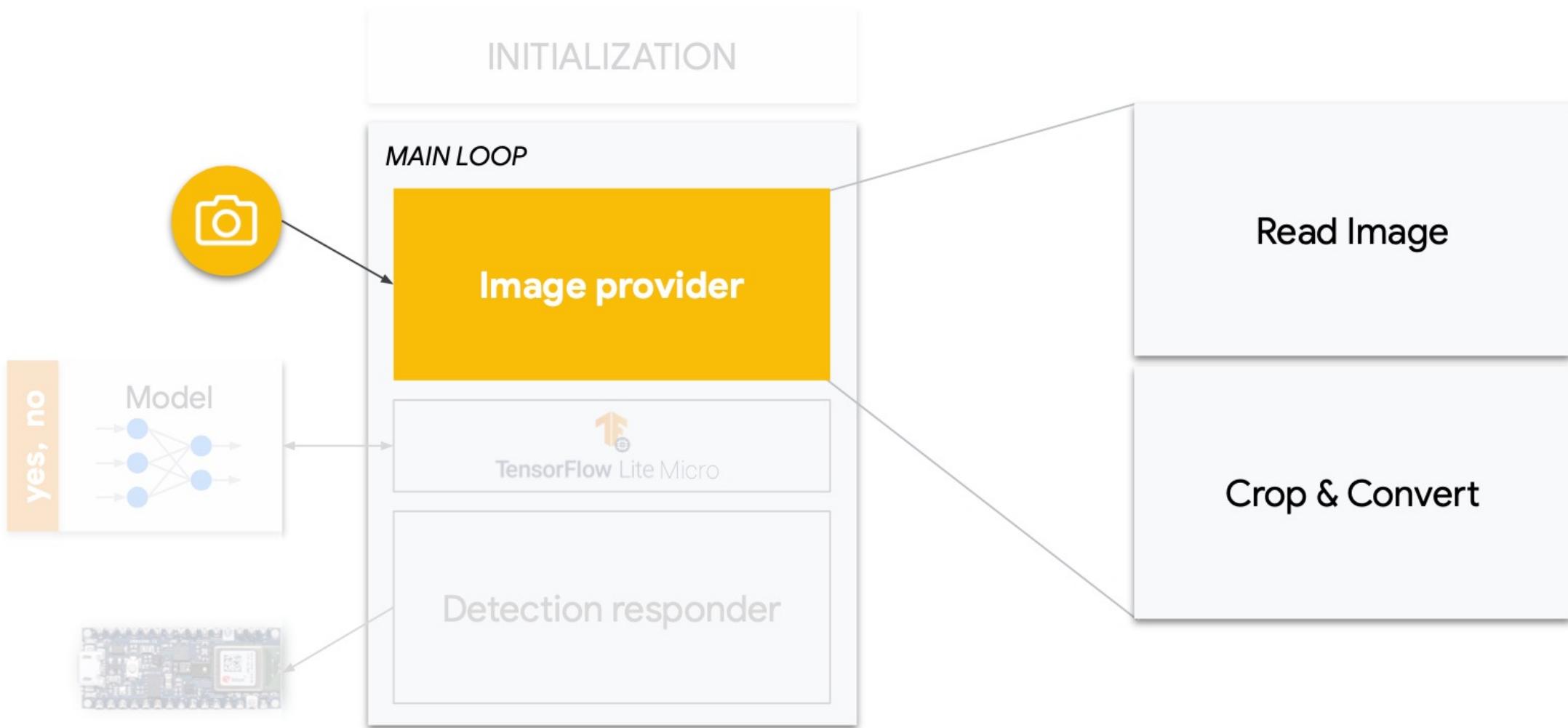
# Initialization



# Initialization



# Pre-processing



# Pre-processing

Read Image

Crop & Convert



144



176

```
// Get an image from the camera module
TfLiteStatus GetImage(tfLite::ErrorReporter* error_reporter,
                     int image_width, int image_height, int channels,
                     int8_t* image_data)
```

# Pre-processing

Read Image

Crop & Convert



QCIF

144



176

```
// Read camera data  
Camera.readFrame(data);
```

# Pre-processing

Read Image

Crop & Convert



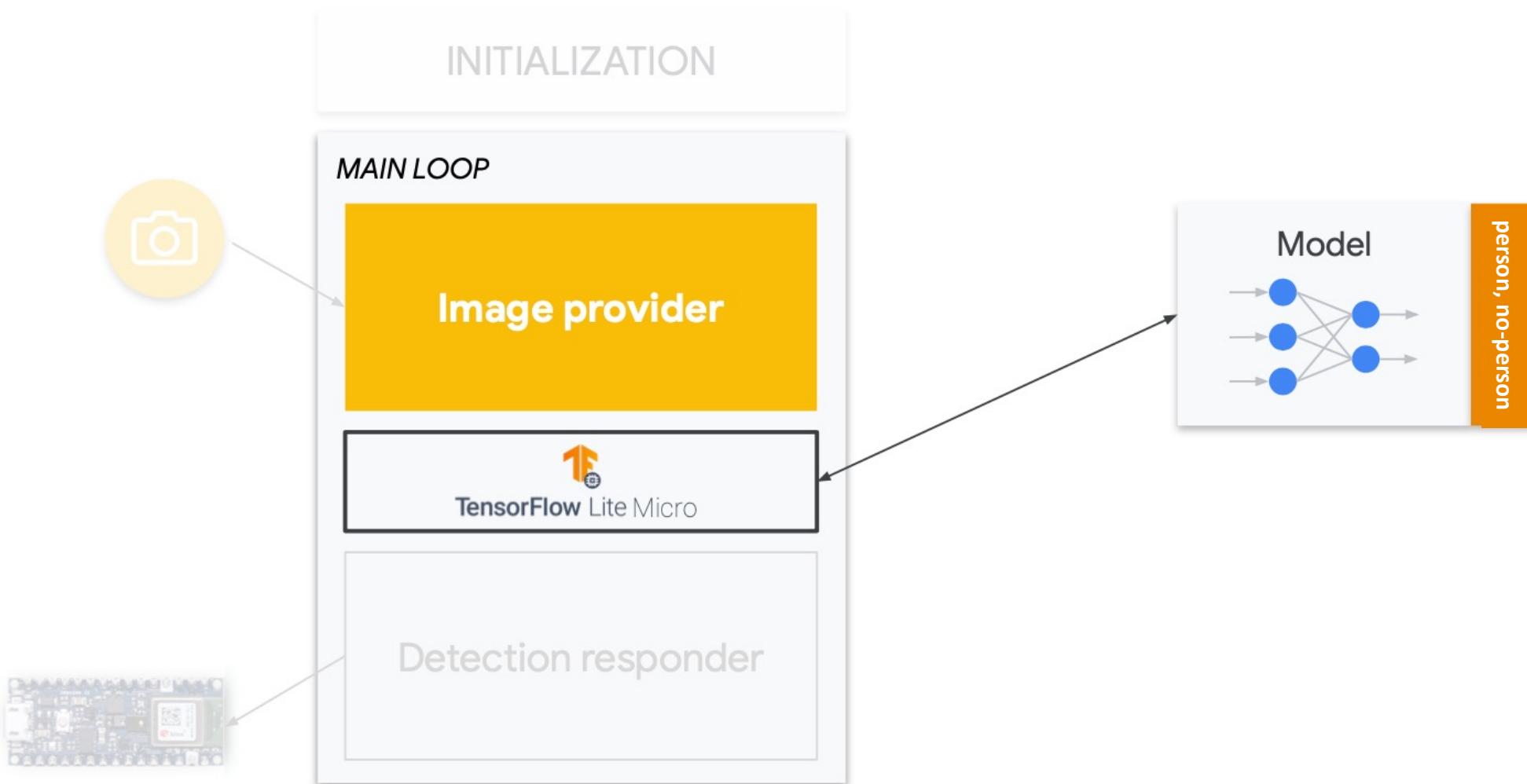
144



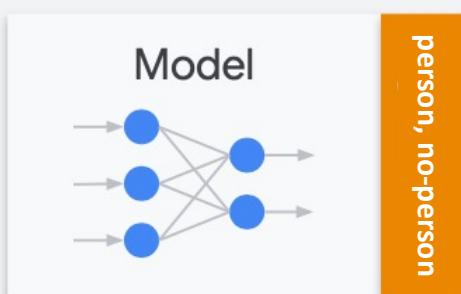
96

```
int min_x = (176 - 96) / 2;  
int min_y = (144 - 96) / 2;  
int index = 0;  
  
// Crop 96x96 image. This lowers FOV, ideally we should downsample  
for (int y = min_y; y < min_y + 96; y++) {  
    for (int x = min_x; x < min_x + 96; x++) {  
        image_data[index++] = static_cast<int8_t>(data[(y * 176) + x] - 128);  
        // convert TF input image to signed 8-bit  
    }  
}
```

# Interpreter + Model

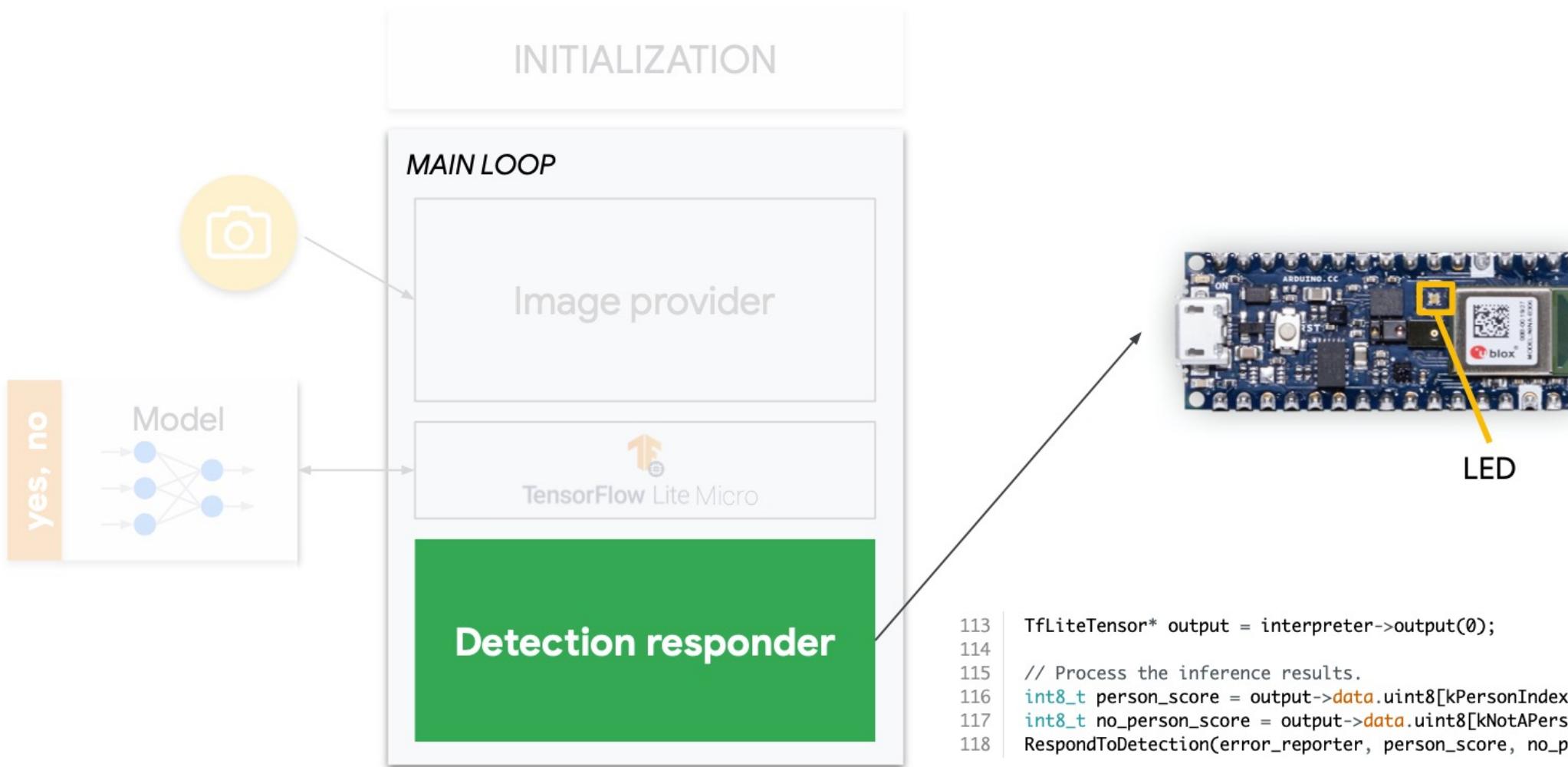


# Interpreter + Model



```
kTfLiteOk != vww_interpreter->Invoke()
```

# Post-processing



## Detection responder

```
if (person_score > no_person_score) {  
    digitalWrite(LEDG, LOW);  
    digitalWrite(LEDR, HIGH);  
} else {  
    digitalWrite(LEDG, HIGH);  
    digitalWrite(LEDR, LOW);  
}
```

## Detection responder

```
if (person_score > no_person_score) {  
    digitalWrite(LEDG, LOW);  
    digitalWrite(LEDR, HIGH);  
} else {  
    digitalWrite(LEDG, HIGH);  
    digitalWrite(LEDR, LOW);  
}
```

# Reading Material

# Main references

- [Harvard School of Engineering and Applied Sciences - CS249r: Tiny Machine Learning](#)
- [Professional Certificate in Tiny Machine Learning \(TinyML\) – edX/Harvard](#)
- [Introduction to Embedded Machine Learning \(Coursera\)](#)
- [Text Book: "TinyML" by Pete Warden, Daniel Situnayake](#)

I want to thank [Shawn Hymel](#) and [Edge Impulse](#), [Pete Warden](#) and [Laurence Moroney](#) from Google and specially Harvard professor [Vijay Janapa Reddi, Ph.D.](#) student [Brian Plancher](#) and their staff for preparing the excellent material on TinyML that is the basis of this course at UNIFEI.

The IESTI01 course is part of the [TinyML4D](#), an initiative to make TinyML education available to everyone globally.

**Thanks**  
And stay safe!

