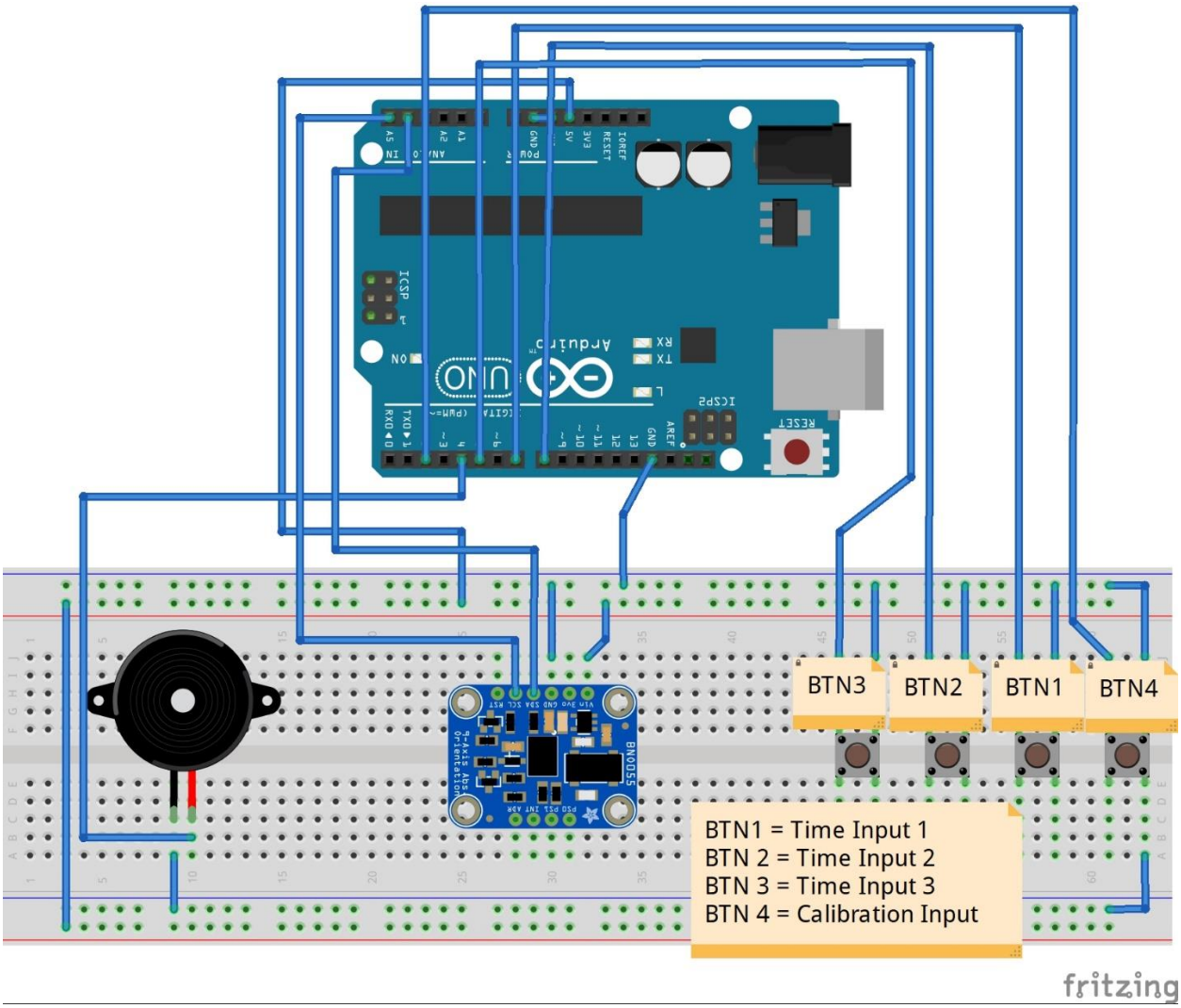
Schematic:


BTN3
BTN2
BTN1
BTN4
BTN1 = Time Input 1
BTN 2 = Time Input 2
BTN 3 = Time Input 3
BTN 4 = Calibration Input
fritzing

Code:

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BNO055.h>
#include <utility/imumaths.h>
#include <Bounce2.h>



/***********************************
    pre-defined configurations
***********************************/
#define buzzer 4
#define b1 2  // button for calibration
#define b2 7  // button to set the mode
#define b3 8  // button to set the mode
#define b4 5  // button to set the mode
#define BTN_DEBOUNCE_INTERVAL (25)
/* Set the delay between fresh samples */
#define BNO055_SAMPLERATE_DELAY_MS (100)



/***********************************
   global variables. These variables will
   occupy some RAM constantly.
***********************************/
int timeButtonState = 1; // initial state set to 1
float temporaryAngle = 50.0;
float T = 0.0;
bool flag = false;

/***********************************
   button debouncers. input will be read
   through these objects
***********************************/
Bounce calBtn; // calibration button debouncer
Bounce timBtn1, timBtn2, timBtn3; // time state input button bouncers

// get a new sensor event
sensors_event_t event;

// BNO055 instance
Adafruit_BNO055 bno = Adafruit_BNO055(55);
```

```
/*********************************************************************
  Function    : CheckTimeButton
  Description : Check if Button 1 is pressed and released. This eleminates
                accidental inputs.
  Parameters  : none
  Returns     : true if button was pressed
*********************************************************************/
bool IsTimBtn1Pressed() {
  timBtn1.update(); // clear any previous state
  if (timBtn1.fell()) { // button is pressed
    do {
      timBtn1.update();
    } while (!timBtn1.rose()); // wait for button release
    return true;
  }
  return false;
}

/*********************************************************************
  Function    : CheckTimeButton
  Description : Check if Button 2 is pressed and released. This eleminates
                accidental inputs.
  Parameters  : none
  Returns     : true if button was pressed
*********************************************************************/
bool IsTimBtn2Pressed() {
  timBtn2.update(); // clear any previous state
  if (timBtn2.fell()) { // button is pressed
    do {
      timBtn2.update();
    } while (!timBtn2.rose()); // wait for button release
    return true;
  }
  return false;
}

/*********************************************************************
  Function    : CheckTimeButton
  Description : Check if Button 3 is pressed and released. This eleminates
                accidental inputs.
  Parameters  : none
  Returns     : true if button was pressed
*********************************************************************/
bool IsTimBtn3Pressed() {
  timBtn3.update(); // clear any previous state
  if (timBtn3.fell()) { // button is pressed
    do {
      timBtn3.update();
    } while (!timBtn3.rose()); // wait for button release
    return true;
  }
  return false;
}
```

```
/*********************************************************************
  Function    : CheckTimeButton
  Description : It checks the status of the three time buttons to determine the time
                button state
  Parameters  : none
  Returns     : void
*********************************************************************/
void CheckTimeButton() {
  if (IsTimBtn1Pressed()) {
    timeButtonState = 1;
  } else if (IsTimBtn2Pressed()) {
    timeButtonState = 2;
  } else if (IsTimBtn3Pressed()) {
    timeButtonState = 3;
  }

  // print button state to serial console
  Serial.println(timeButtonState);
}

/*********************************************************************
  Function    : UpdateAngle
  Description : It updates values of variable with latest z-axis angle and displays
                angle on serial monitor.
  parameters  : none
  Returns     : void
*********************************************************************/
void UpdateAngle() {
  bno.getEvent(&event);
  T = event.orientation.z;
  Serial.print("\tZ: ");
  Serial.print(event.orientation.z, 4);

  // new line for the next sample
  Serial.println("");

  // wait the specified delay before requesting next data
  delay(BNO055_SAMPLERATE_DELAY_MS);
}
```

```
/*********************************************************************
  Function    : CheckButton1 renamed to CheckCalibrationButton()
  Description : It checks the button 1 state (calibration button) if button is pressed
                then the function calibrates the value of the temporaryAngle variable.
  Method      : Once the calibration button is held down until the serial monitor
                shows 20, the calibrated angle will keep displaying onto the screen
                after you let go of the button. Then, once you click it down again,
                the temp value will be updated with whatever the last value was
                immediately before the button was clicked.
  Parameters  : none
  Returns     : void
*********************************************************************/
void CheckCalibrationButton()
{
  calBtn.update(); // clear previous state
  if (calBtn.fell()) { // cal btn pressed
    int count = 0;
    do {
      Serial.print(++count);
      Serial.print(" "); // a space between 2 digits
      calBtn.update(); // must clean previous state to get new state
    } while (count < 20 && !calBtn.rose());
// either count reaches 20 or button is released

    if (count < 20) { // calibration button was released earlier
      return;
    }

    Serial.println("");
    Serial.print("\tCount has exceeded 20");
    Serial.println("");

    // calibrate temporary angle and print it to serial monitor
    do {
      bno.getEvent(&event);
      temporaryAngle = event.orientation.z;
      Serial.print("\tCalibrated Z Angle: ");
      Serial.print(temporaryAngle);
      Serial.println("");
      delay(BNO055_SAMPLERATE_DELAY_MS); // delay before next event data
      calBtn.update(); // clean previous state to get new state
    } while (!calBtn.fell());

    // wait for button release
    do {
      calBtn.update();
    } while (!calBtn.rose());
  }
}
```

```
/*********************************************************************
  Function    : PlayTone()
  Description : It plays the buzzer for about 100+ milliseconds
  Method      : Plays the buzzer at a specific tone and duration of buzzer sound can
                be changed
  Parameters  : none
  Returns     : void
*********************************************************************/
void PlayTone() {
  tone(buzzer, 262, 1000 / 8);
  for (int i = 0; i <= 200; i++)
  {
    delay(1);
  }
}

bool ConditionAppliesFor(unsigned long dur) {
  unsigned long startTime = millis();
  unsigned long endTime = startTime;
  do {
    UpdateAngle();
    endTime = millis();
  } while ((endTime - startTime) < dur && T < temporaryAngle);

  return (endTime - startTime) >= dur;
}

void setup() {
  // setup button debouncers first
  calBtn.attach(b1, INPUT_PULLUP);
  timBtn1.attach(b2, INPUT_PULLUP);
  timBtn2.attach(b3, INPUT_PULLUP);
  timBtn3.attach(b4, INPUT_PULLUP);

  // setup debouncers' interval
  calBtn.interval(BTN_DEBOUNCE_INTERVAL);
  timBtn1.interval(BTN_DEBOUNCE_INTERVAL);
  timBtn2.interval(BTN_DEBOUNCE_INTERVAL);
  timBtn3.interval(BTN_DEBOUNCE_INTERVAL);

  // setup the buzzer
  pinMode(buzzer, OUTPUT);

  // setup the serial console output
  Serial.begin(9600);
  Serial.println("Orientation Sensor Test");
  Serial.println("");

  // initialize the sensor
  if (!bno.begin()) {
    // there was a problem detecting the BNO055 ... check your connections
    Serial.print("Ooops, no BNO055 detected ... Check your wiring or I2C ADDR!");
    while (1);
  }
```

```
  delay(1000);

  bno.setExtCrystalUse(true);
}

void loop() {
  // see if user pressed the calibration button to calibrate temporary angle
  CheckCalibrationButton();

  // see if user pressed any timer state button
  CheckTimeButton();

  // update gyro state
  UpdateAngle();

  // check if T is greater than the calibrated value and flag state is false, start
the main task
  if (T <= temporaryAngle && !flag) {
    flag = true;

    // perform beeping according to the state
    if (timeButtonState == 1) { // if state == 1, immediate beeping is enabled
      PlayTone();
    } else if (timeButtonState == 2) { // if state == 2, wait 2 seconds before beeping
      // watch 2 seconds for condition
      if(ConditionAppliesFor(2000)) {
        PlayTone();
      }
    } else if (timeButtonState == 3) { // if state == 3, wait 4 seconds before beeping
      if(ConditionAppliesFor(4000)) {
        PlayTone();
      }
    }
  } else if (T > temporaryAngle) {
    flag = false;
  }
}
```