Interdisciplinary school project using Nintendo Wii controller for measuring car speed

Nils Kristian Hansen and James Robert Mitchell
University of Agder, P.O. Box 422, 4604 Kristiansand, Norway
E-mails: nils.k.hansen@uia.no james.r.mitchell@uia.no

Abstract
This work examines the feasibility of employing a Nintendo Wii game controller for measuring car speed in an interdisciplinary school project. It discusses the physical characteristics of the controller and of vehicle headlights. It suggests how an experiment may be linked to topics in mathematics, statistics, physics and computer science. An algorithm for calculating speed from repeated recordings of car headlights is provided. Finally the results of repeated experiments with an approaching car are provided.

Introduction

During the last four years several articles have been published on the subject of using a Nintendo Wii remote controller (Wiimote) for physics experiments, making use of its built in accelerometers and IR camera. Vannoni and Straulino [1] use it for analysing pendulum motion. Ochoa, Rooney and Somers [2] employ it in investigating harmonic motion on a string. Wheeler [3] recapitulates these experiments and expands them by presenting tailor made software and introducing experiments using multiple Wiimotes and the Wii Balance Board.

These articles all aim mainly at demonstrating how Wiimotes can replace commercial data loggers, and use a black box approach to the software. This work however, demonstrates how a Wiimote can be used to perform an outdoor experiment not achievable by commercial data loggers. It uses a white box approach to the software, facilitating an interdisciplinary student project, joining topics from physics, mathematics and computer science.

It is based on the fact that ordinary tungsten car headlights emit near-IR, thus ought to be detectible by the Wiimote IR camera. When the distance between the headlights of private cars is known, as well as the focal length of the Wiimote camera, the distance of a car can be calculated using similar triangles. By repeated measurements, the average speed of an approaching car can be determined and displayed.

IR-sources, types of IR, elementary remote sensing and more generally radio wave communication, such as the Wiimote Bluetooth communication, are topics that may be linked to this experiment. Programming skills may be challenged through developing the GlovePIE script, where the use of a finite state machine is recommended. Application of mathematics is required in calculating distance by Pythagoras and similar triangles, and computing average speed from recorded distances. A lesson may also be complemented through production of charts of statistical data, and the working limitations of the equipment open for discussion in project work.

Wiimote characteristics

A Wiimote is a video game control device, communicating with the gaming console by Bluetooth.

Its characteristics are not published by the manufacturer, but Vannoni and Straulino [1] states that it has a three-axes accelerometer, measuring along three perpendicular axes over a range of ± 3g with 10 % sensitivity, and that it has a 1024 x 768 pixel camera with an IR filter in front, tracking up to 4 IR sources.

A test with commercially available IR diodes with frequencies of 850 nm, 880 nm, 940 nm and 950 nm respectively demonstrated the Wiimote camera to be most sensitive to 940 nm.

A test with a 940 nm IR diode powered by 50 mA at a distance of 1 metre yielded a sensitivity angle of 42° horizontally and 30° vertically.

The average of 10 readings at distances from 16 to 43 cm using two IR diodes with an internal distance of 9.6 cm indicated the focal length of the camera to be 1328 +/-5 pixels.

A Wiimote with top lid removed is shown in figure 1.

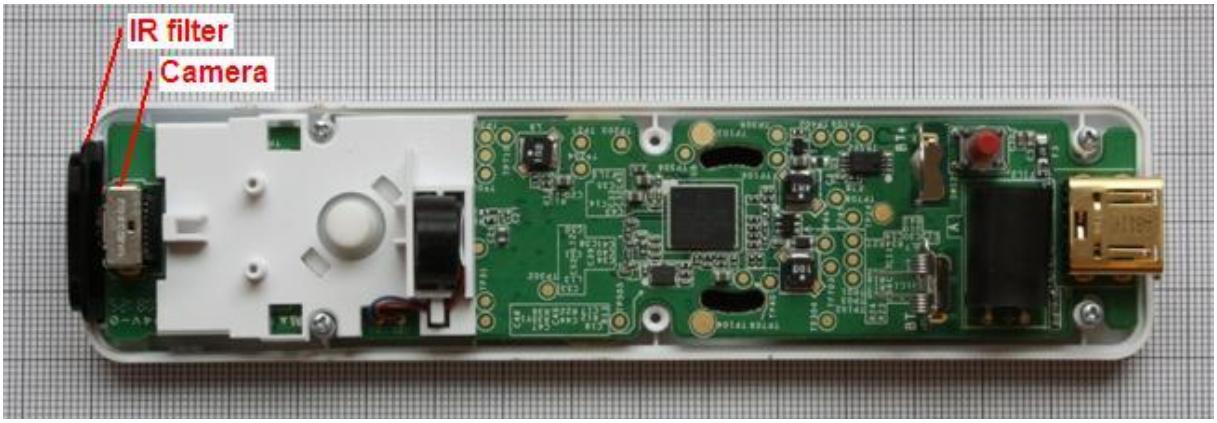

Figure 1 Wiimote with top lid removed.

Vehicle characteristics

A sample of 50 random private cars showed the average distance between the headlight bulbs to be 110 cm, with a relative standard deviation of 9 %.

Software

The Wiimote reports the position of IR-sources in its internal 1024 x 768 pixel grid. To convert the position of car headlights into the distance of an approaching car, a software interface is needed. Karl Kenner [4] provides GlovePIE [5], a programmable input emulator with an interface as shown in figure 2.

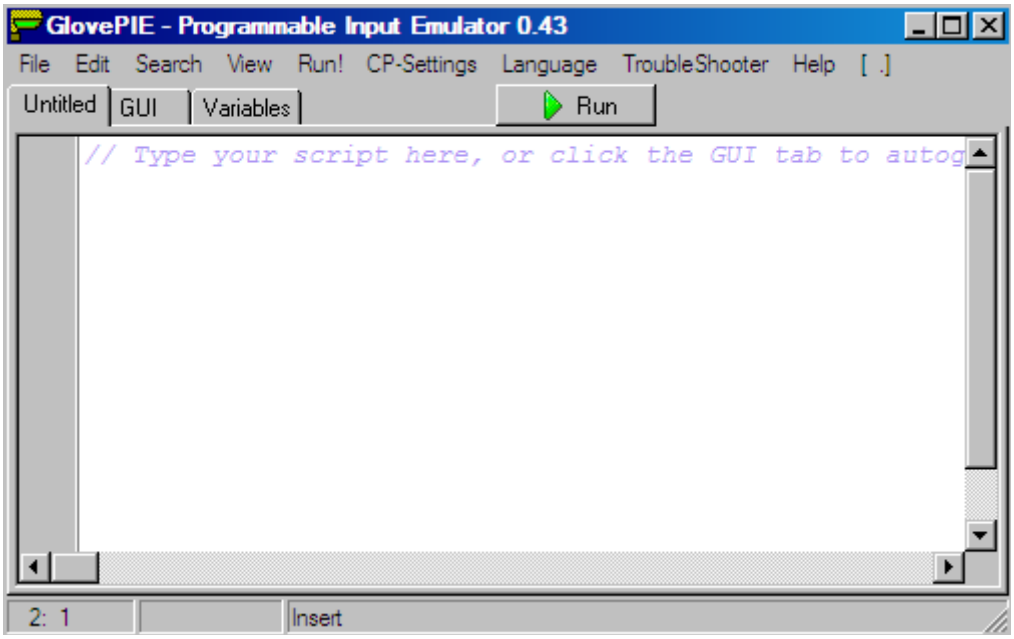

Figure 2 GlovePIE interface.

GlovePIE scripts execute in infinite loops.

GlovePIE employs a proprietary script language, but accepts a variety of well-known programming syntaxes [6]. This work uses Java syntax.

GlovePIE specific codes employed in this work are:

- *var*. Variable prefix. Prefix is separated from variable by a dot.
- *timeStamp*. System clock. Unit is seconds with two decimals followed by the text "Seconds". This text may be removed by dividing by one second. (1s)

- *wiimote.dotNvis*, $N \in \{1,2,3,4\}$. True if at least N IR-sources are detected.
- *wiimote.dotNX*, $N \in \{1,2,3,4\}$. X-coordinate of IR-source N. Unit is pixels.
- *wiimote.dotNY*, $N \in \{1,2,3,4\}$. Y-coordinate of IR-source N. Unit is pixels.
- *debug =*. Outputs what follows the equal sign to a small window in the GlovePIE interface.
- *starting*. Built-in variable. *true* in first loop, *false* otherwise.

Calculating distance and speed

The mathematics of calculating the distance to a pair of IR sources is simple, making use of Pythagoras and similar triangles.

In GlovePIE the position of two IR sources is made available as two coordinate pairs. Denoting them (x_1, y_1) and (x_2, y_2), their relative distance in pixels, a, may be calculated by Pythagoras:

$$a = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Knowing a, as well as the focal length of the Wiimote, b, and the real world distance between the IR sources, c, the range between the IR sources and the Wiimote, d, may be calculated by similar triangles:

$$\frac{d}{b} = \frac{c}{a} \Rightarrow d = \frac{bc}{a}$$

This is illustrated in figure 3. The units of a and b are pixels. For c and d this work uses metres.

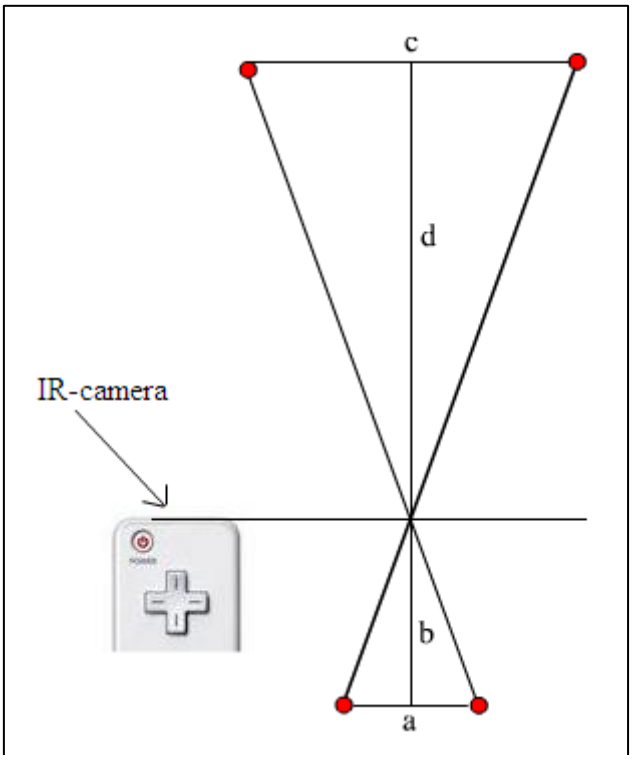

Figure 3 Calculating distance using similar triangles.

With b set to the Wiimote focal length of 1328 pixels and c to the average distance between car lights to 1.1 metres, the resulting equation is:

$$d = \frac{1328\,px \cdot 1.1\,m}{a\,px} \approx \frac{1461}{a} m$$

Instant speed is calculated as distance covered divided by time elapsed between two readings:

$$s_n = \frac{d_{n-1} - d_n}{t_n - t_{n-1}}$$

Average speed is calculated by averaging all samples of instant speed:
$$\bar{s} = \frac{\sum s_n}{n}$$

Average speed calculation algorithm
During algorithm design it had to be taken into consideration that

- GlovePIE provides no array structures, thus storing speed samples and performing calculations and corrections at the end would be impractical. All calculations would have to be performed on the fly.
- If track of headlights was lost, the algorithm should reset.
- Headlights detected too close to yield sufficient speed samples should be ignored.
- The algorithm should be easy for students to understand and modify.

Initial experiments showed that headlight tracking was stable between 37 and 6 metres. Thus, allowing a margin of error, the algorithm was designed to start sampling when first reading was in the range of 35 and 25 metres and end with first reading below 8 metres. The algorithm is a finite state machine with three states, execution starting in state *waiting*:

State *waiting*:
  If two IR-sources detected and distance is $25m < d < 35m$:
    Sample initial time t_0 and distance d_0.
    Set initial accumulated speed to $c_0 = 0$.
    Switch to state *sampling*.
  Else:
    Do nothing.
State *sampling*:
  If two IR-sources detected and distance is $6m < d < 35m$:
    Sample time t_n and distance d_n.
    Calculate instant speed $s_n = \frac{d_{n-1} - d_n}{t_n - t_{n-1}}$.
    Calculate accumulated speed, $c_n = c_{n-1} + s_n$.
    If distance $d < 8m$.
      Calculate average speed $s = \frac{c_n}{n}$.
      Switch to state *displaying*.
  Else:
    Do nothing.
State *displaying*:
    Display average speed, s, in unit desired, e.g. $s \cdot 3.6\ km\ h^{-1}$

The corresponding GlovePIE script code can be found in appendix A. It also includes a timer forcing a switch back to state *waiting*, thus resetting the system:

- After 150 loops in state *displaying*
- After 30 loops in state *sampling* without a proper reading.

Practical considerations and limitations
Initial experiments showed that on clear days, ubiquitous solar reflections throw the IR-camera severely off, thus experiments can only be carried out on overcast days or after sunset. Also fog reduced the headlight detection range.

Another problem is that some headlights employ xenon bulbs instead of tungsten, and are undetected by the Wiimote far IR camera. Also headlight reflectors reflect a broad beam of light forward, which may incur a somewhat unpredictable intensity profile relative to viewing angle. The

Wiimote software may therefore vary the detected position of the headlight centre point according to the precise vehicle direction. Although a possible source of noise, this should average out over continuous measurements of vehicle speed.

Vehicles with deviating distance between the headlights will yield incorrect readings. This will be a major problem with buses, lorries and vans, and readings on these should be ignored.

Further concerns are the quantized nature of pixels at the limit of resolution and the timestamp of the software. Subsequent readings may be associated with the same pixel pair, yielding a speed of zero. Time lapse between timestamps in the software alternates between 30 ms and 40 ms.

However recordings of an approaching car showed that though instant speed samples were erratic, the average quickly stabilized and the relative standard deviation dropped and stayed below 10 %.

The system refuses to report speeds above approximately 60 km h^{-1}. It may however be possible to remedy this by modifications to the software.

Practical experiment

An experiment was carried out by installing the Wiimote on a tripod at the roadside and driving by 6 times at 20, 30, 40 and 50 km h^{-1} respectively and recording the reported speed. The headlight separation for this vehicle was 1,0 m. Weather conditions were full daylight but overcast. The results are shown in table 1.

Speedometer	20	30	40	50
Average of 6 Wii readings	18	28	37	47
Relative standard deviation of Wii readings	4 %	4 %	4 %	5 %

Table 1: Correspondence between speedometer and Wii at different speeds (km h^{-1}).

Please note that this is not intended as a precise assessment of the precision of the Wee recordings, there are several error sources. Although the test car was driven as close to constant speed as possible, the lag in speedometer reaction and driving accuracy may account for some of the relative standard deviation. Neither is it given that the speedometer reading is the true speed of the vehicle.

However the results may be interpreted as an indication of what may be expected in a school project.

Conclusions and further work

This work demonstrates that it is technically feasible to employ a Wiimote for estimating the speed of cars based on IR-radiation from the headlights, within certain working limitations. Even though precision is limited, an experiment shows promising results.

The equipment is furthermore cheap, robust, portable and easy to use, and the software required can be developed by students.

Further work is left to anyone who wants to pick up on the white box approach of this work.

References

[1] Vannoni M and Straulino S 2007 Low-cost accelerometers for physics experiments *European Journal of Physics* **28** 781-7

[2] Ochoa R, Rooney F G and Somers W J 2011 Using the Wiimote in Introductory Physics Experiments *Physics Teacher* **49** 16-18

[3] Wheeler M D 2011 Physics experiments with Nintendo Wii controllers *Physics Education* **46** 57-63

[4] http://glovepie.org/glovepie.php

[5] http://glovepie.org/lpghjkwer.php

[6] http://glovepie.org/w/index.php?title=Preliminary_Documentation

Appendix A, GlovePIE script code

```
if( starting ){
  var.state = "waiting"; // Initial state
}
// Code common to two states
if( var.state == "waiting" || var.state == "sampling" ){
  // If two IR sources are detected
  if( wiimote.dot1vis && wiimote.dot2vis ){
    // Use Pythagoras to calculate distance between IR-sources
    var.x_dist = wiimote.dot2X - wiimote.dot1X;
    var.y_dist = wiimote.dot2Y - wiimote.dot1Y;
    var.a = sqrt( sqr( var.x_dist ) + sqr( var.y_dist ));

    // Use similar triangles to calculate distance to IR-sources
    var.d = 1461 / var.a;
  } else {
    var.d = -1; // No distance available
  }
}
// Act as proper in current state
if( var.state == "waiting" ){
  debug = "Waiting ...";
  var.loops = 0;
  // If distance in "first detect" range, set initial values
  if( 25 < var.d && var.d < 35 ){
    var.count = 0;
    var.tOld = timeStamp / 1s;
    var.dOld = var.d;
    var.accSpeed = 0;
    var.state = "sampling";
  }
}
if( var.state == "sampling" ){
  debug = "Sampling ...";
  if( 6 < var.d && var.d < 35 ){ // If distance in range, calculate instant speed
    var.loops = 0;
    var.count++;
    var.tNew = timeStamp / 1s;
    var.dNew = var.d;
    var.iSpeed = ( var.dOld - var.dNew ) / ( var.tNew - var.tOld );
    var.accSpeed += var.iSpeed;
    var.tOld = var.tNew;
    var.dOld = var.dNew;
    if( var.d < 8 ){ // If distance in stop range, calculate avg. speed
      var.avgSpeed = var.accSpeed / var.count;
      var.state = "displaying";
    }
  } else {
    var.loops++; // Count of loops without proper sample
    if( var.loops >= 30 ){ // No sample for 30 loops, so car is probably lost
      var.state = "waiting";
    }
  }
}
if( var.state == "displaying" ){
```

```
  debug = "Speed: " + round( var.avgSpeed * 3.6 ) + " km h-1";
  var.loops++ // Count of loops while displaying
  if( var.loops >= 150 ){ // Speed displayed long enough
    var.state = "waiting";
  }
}
```