A wireless marker system to enable evoked potential recordings using a wireless EEG system (EPOC) and a portable computer

Johnson Thie

School of Electrical and Information Engineering

The University of Sydney, Australia

Abstract

As wireless EEG devices have become affordable at low cost, have a small form factor and quick setup time, they can be deployed at universities and schools for teaching purposes. However they have not been applied for evoked potential recording since they lack an option to receive stimulus markers. Meanwhile evoked potential recording is required for functional assessment of the sensory systems such as auditory and visual. This paper describes a wireless system that embeds information about the stimulus in the EEG channels. The transmitter unit is connected to the stimulus device to detect the stimulus and transmit the stimulus information to the receiver unit. The receiver unit attached to two of the EEG electrodes decodes the information and generates a pulse across the electrodes. The pulse width conveys the information about the stimulus. Hence the stimuli are synchronised with the EEG data allowing users to evaluate the evoked potentials in the offline processing. The wireless marker system was verified with audio stimuli consisting of 1000Hz and 1200Hz tones and reliably generated pulses with 100ms and 200ms width respectively. The delay between the onset of the tone and the onset of the pulse was 19.3 ± 0.1ms. Since the variability of the delay was under 1ms and so negligible, the evoked potentials could be evaluated reliably. The evoked potential could be shifted back by 19.3ms to compensate for the delay. The system was also verified with a black-and-white checkerboard pattern stimuli and reliably generated pulses with 100ms width when the pattern reversed. The delay between the onset of the reversal and the onset of the pulse was 6.4ms. Similarly the variability of the delay was negligible.

Index Terms

EEG, Emotiv, EPOC, evoked potential, Arduino

I. Introduction

Wireless devices for recording physiological signals have become available for general consumer usage at a low cost. Recently wireless devices to record EEG were commercialized by Neurosky and Emotiv for consumer applications, particularly in video games. Users are provided with the software development kit that allows them to build their own software applications, e.g. controlling an avatar's action using their thoughts or practising meditation to elevate the alpha wave (burst of 10Hz oscillations). Neurosky and Emotiv set up an online store for users to offer their applications to public. Users also have access to the raw EEG and so can learn how to analyse the EEG. Similarly they can build custom applications which directly use the raw EEG.

The devices are suitable candidates for deployment at universities and schools for demonstrating brain functions and hands-on applications of engineering and programming in biology at a much lower cost. For example, students will not only be able to observe the presence of alpha waves which indicate the alertness level but also develop custom signal processing and pattern recognition algorithms to quantify the alertness level. In addition, more students will be able to develop applications for brain computer interface and compete at the annual competition.

To date, these devices have been primarily used for research. Examples include using the Emotiv's device to record steady-state visual evoked potential [1] and P300 potential [2], [3]. The Neurosky's device has also been used to record steady-state visual evoked potential [4], deduce a person's emotion [5] and control a wheelchair [6]. In the Emotiv's developer forum, there have been discussions on using the device for school projects (e.g. http://emotiv.com/forum/messages/forum4/topic935/message5585/#message5585 and http://emotiv.com/forum/messages/forum15/topic649/message3721/#message3721).

Another important aspect of brain functions is the sensory response. For instance, the auditory response can be obtained when the subject's EEG is recorded while listening to certain audio stimuli. Students can then learn the behaviour of the auditory and visual cortices in response to various audio and visual stimuli respectively as well as writing programs to evaluate and analyse the evoked potentials. They can explore various algorithms to analyse and quantify the evoked potentials, build a database of evoked potential parameters and devise an algorithm to automatically detect abnormal evoked potentials.

Recently, Babcock et al. [7] demonstrated that the Emotiv's device can be used to record

auditory evoked potentials. They showed that the evoked potentials were comparable to those obtained using a medical-grade EEG device such as Neuroscan. The same setup can also be adapted for visual evoked potentials.

One significance in Babcock et al.'s work was the ability to synchronise the EEG recording from the Emotiv's device with the stimuli. The absence of synchronisation was a major shortfall in the Emotiv's device. Thie [8] developed the marker circuit for the Emotiv's EEG device that embedded the stimulus onset in the EEG channels thereby synchronizing them. The circuit produced a pulse on two of the EEG electrodes when an auditory stimulus was detected. The pulse had the same duration as the stimulus and a fixed amplitude of about 300µV. Since the pulses representing the stimulus onset were embedded with the EEG, the EEG could be divided into epochs that were aligned to the pulses. Then the evoked potential was evaluated by averaging the epochs. Unfortunately the marker circuit used wires to connect to the EEG device and the computer that generated the stimuli. Hence the overall system became no longer wireless.

This paper describes a wireless system to embed the stimulus onset to the EEG channels such that the overall system remains wireless. The transmitter unit is attached to the computer that produces the stimuli, detects the stimulus onset and sends the corresponding information to the receiver unit via an infrared link. The receiver unit is mounted on the headset and generates pulses on two of the EEG electrodes. The width of the pulses are configurable to represent different types of stimulus. The system accommodates for both auditory and visual evoked potentials. Though it was designed to work with Emotiv's EEG device, it should readily work with other wireless EEG devices too. This system offers several advantages over Thie's marker circuit [8], namely, (a) the overall system remains wireless, (b) it can discriminate between different tones (audio frequencies) and in turn produce pulses with different widths, (c) it can accommodate for auditory and visual evoked potentials. For convenience, the system will be referred to as "wireless marker" and the pulses as "marker pulses".

The wireless marker was designed to be low cost, customisable, built using widely available components and open source in order to support the application of the wireless EEG devices for educational purposes. Given these constraints, the transmitter and receiver units employ off-the-shelf Arduino boards which meet all the criteria above. The code editor and compiler can be downloaded for free. The interface circuits, often called "shield" by the Arduino community, used generic electronic components which can be purchased from major electronic

vendors such as Farnell and Digikey. The schematics and board layout files are available upon request.

Section II provides the technical description of the transmitter and receiver units of the wireless marker and the methods to verify their usage in auditory and visual evoked potential recording. Section III describes the verification results. Section IV concludes the system design and verification. Appendices A and B outline the transmitter and receiver code respectively.

II. Methods

A. *Transmitter*

The transmitter unit consists of an Arduino Uno board and a shield. The shield's schematics is shown in Figure 1. The shield board was designed such that it could be mounted on top of the Arduino board through the headers, PA1 to PA4. Details about the Arduino Uno board can be found in http://www.arduino.cc/. The transmitter has three operating modes, namely, analogue, digital and serial which are selected using a slide switch (S1).

In the analog mode, the shield expects an audio signal from a 3.5mm socket (J2), removes its DC offset and amplifies the signal by $10\times$. Such a high gain ensures that the peak of the audio signal will reach the supply rail of 5V. A 2.5V reference is used to bias the amplifier since it operates on a single rail power supply. Then the Arduino microcontroller samples the signal through the analog input A0 and determines its frequency through the function `run_audio_input_two_tones()`. As the analog signal oscillates from 0 to 5V, the digital samples will oscillate from 0 to 1023. The onset of the signal is detected when the sample exceeds 560 (`ANATH`). The samples are stored in a buffer and whenever their values increase and cross 511 (`ANAREF`), a counter is incremented. The counter corresponds to the number of oscillations in the buffer. Given a buffer size of 100 samples and a sampling period of about 0.1ms (the typical sampling period of `analogRead()`), the buffer contains 10ms of the signal. Hence given 1000Hz and 1200Hz audio tones, the corresponding counter values should be 10 and 12 respectively. However from experiment, the counter values turned out to be 12 and 14 respectively. Hence the threshold to discriminate between these two frequencies was set to 13. When the lower frequency tone is detected, the pulse width is set to 100ms. Otherwise, the pulse width is 200ms. The pulse width was limited to 250ms and so can be represented as a byte.

In the digital mode, users can use any of the Arduino's digital pins (D2 to D8) which can be accessed through pins 1 to 7 of the connector J3 respectively. When all pins are set as an input, a unique pulse width can be transmitted when any individual pin or any combination of the 7 pins changes its state. Hence up to $2^7 - 1 = 127$ different pulse widths are available.

The digital mode is most likely used to detect changes in the visual stimuli for visual evoked potential recording. The visual stimulus is a black and white checkerboard pattern that reverses its pattern (black to white and white to black) periodically. A phototransistor circuit, shown in Figure 2, is placed on the corner of the monitor screen so that it can detect the reversal. The VCC and Out nodes were connected to pins D2 and D3 respectively where pin D2 is configured as an output and set to high while pin D3 is configured as an input. These pins are linked to pins 1 and 2 of the 8-pin mini DIN connector (J3) respectively.

When a white check is displayed in front of the phototransistor, it will be switched on and its collector voltage drops to a very low voltage. This in turn switches off the MOSFET and pulls its drain voltage to high. Hence a high state is detected at the digital input D3. On the other hand when a black check is displayed, the phototransistor is switched off and its collector voltage is pulled to high. In turn, the MOSFET is switched on pulling the drain voltage to low. Hence a low state is detected. The function `run_vep()` detects the changes at the digital input and sends a pulse width of 100ms when the stimulus reverses its pattern occurs.

In the serial mode, the function `run_serial_input()` reads a byte from the UART port available at the digital pins D0 and D1. The value is assigned to the pulse width directly and sent to the receiver. The UART port can be connected to the PC's serial port via a RS-232 to TTL converter.

B. Infrared link

An infrared (IR) link is used to send the value of the pulse width to the receiver unit. The infrared (IR) link relies on an IR LED (TSAL6200, Vishay semiconductor) and IR receiver (TSOP4P38, Vishay semiconductor). These components are inexpensive and suitable for a short distance and low data rate transmission. The maximum expected transmission distance is about 1m and the data rate is only 2000bit/s. The data transmission starts by sending a start bit (low). Then the data is transmitted serially starting with the least significant bit. No parity check is implemented. The duration of each bit is 0.5ms. So the overall transmission of one byte only takes 4.5ms.

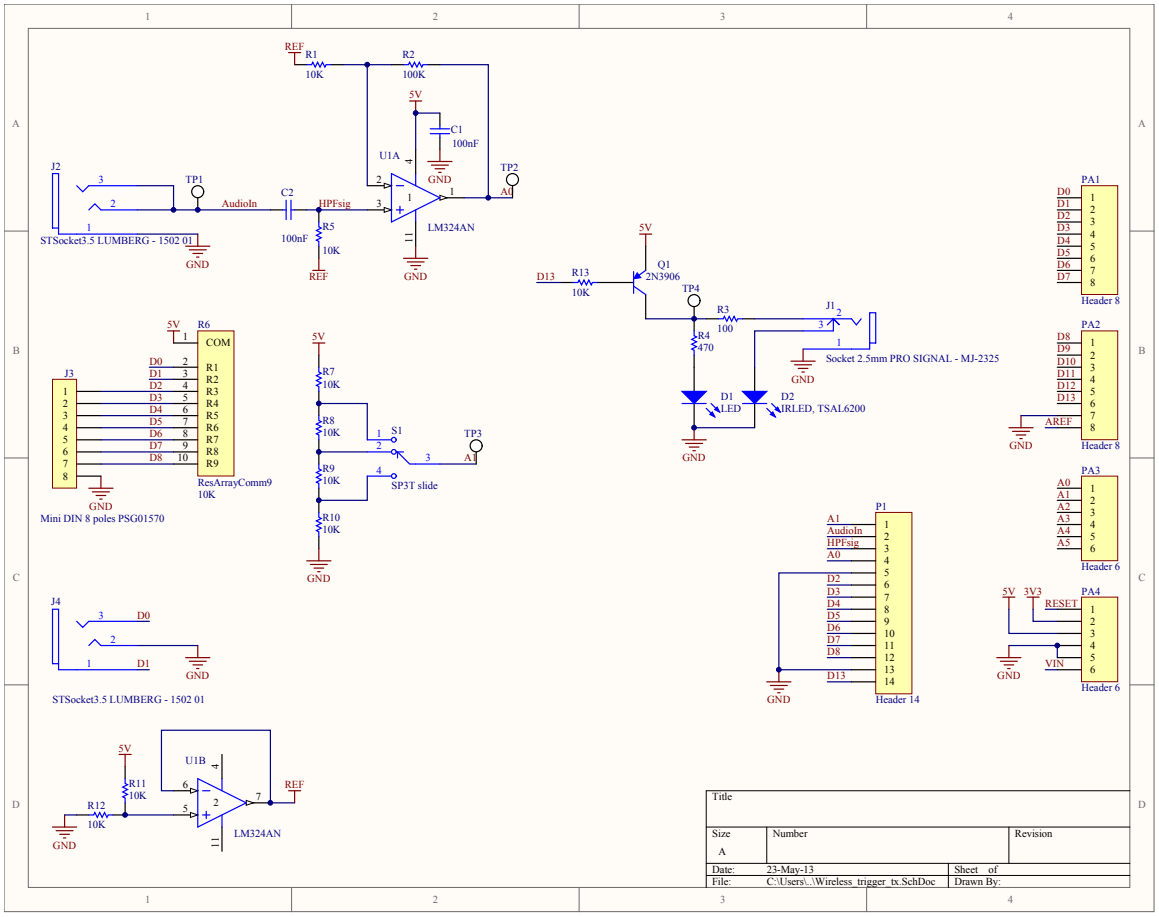

Fig. 1: Schematics of the transmitter shield.

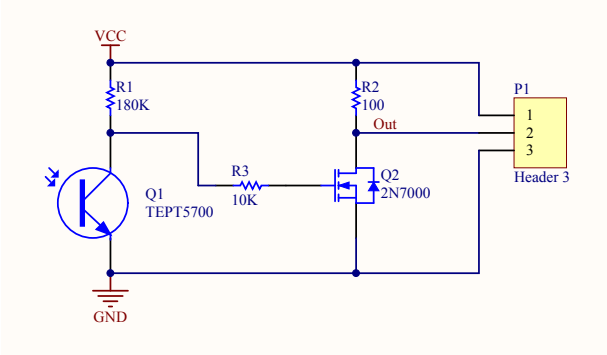

Fig. 2: Schematics of the phototransistor circuit.

The output of the IR receiver in the receiver unit is 3.3V (high) when it does not receive any 38kHz IR signal. When such IR signal is detected, its output switches to 0V (low).

The IR LED in the transmitter unit is driven by a PNP transistor (Q1), 2N3906, which is controlled by the microcontroller's digital output D13. The IR LED is switched on when a 38kHz pulse is produced at D13. The IR LED is programmed to be active low. That is when

the data bit is high, the IR LED is off so that the IR receiver's output is high, and vice versa. A regular LED is also driven by the same transistor as an indicator of the IR transmission.

The 38kHz signal is generated by the Timer2's interrupt. Timer2 is operating in CTC mode with a clock frequency of 16MHz. When its counter reaches 207, it executes the interrupt subroutine which toggles the output at D13. The interrupt subroutine is then called every 13µs. Hence the period of the output signal at D13 is 26µs which is equivalent to 38kHz.

C. Receiver

The receiver unit consists of an Arduino Pro Mini board operating at 3.3V and a shield. The shield's schematic is shown in Figure 3. The headers P1 and P3 attach to the Arduino board. The output of the IR receiver on the shield is attached to the digital pin D9 which is configured as an input. The microcontroller reads the pulse width from the sequence of bits and produces a pulse with the corresponding pulse width on the digital pin D2. The pulse is subsequently attenuated by $10,000\times$ using a voltage divider circuit. The output of the voltage divider is attached to one of the EEG electrodes and the other EEG electrode is attached to the circuit ground through P2. Hence, the EEG device receives a pulse with an amplitude of at most 330µV which is sufficiently low not to damage the EEG device. The receiver unit is powered using a 1/2AA battery which has a typical voltage of 3.6V. For convenience, a regular LED is also attached to D2 to indicate when the pulse is generated. The receiver unit together with its enclosure (1593K, Hammond manufacturing) weighs about 200g and has a dimension of 66mm(L) $\times$ 67mm(W) $\times$ 22mm(H). Hence the unit can be mounted on the EEG headset.

D. Biasing the marker channels

The two electrodes connected to the receiver unit are biased to the DRL electrode in the same way as in [8]. A 4.7kΩ resistor is attached between each of the two electrodes and the DRL electrode. Hence these two electrodes do not need cotton felts inserted into the electrode assemblies.

E. Verification of the analog mode

This section describes the methods to verify that the correct pulse width is transmitted and the delay between the stimulus onset and the pulse onset is constant. A Matlab (Mathworks Inc) code was written to play 1000Hz and 1200Hz audio tones every 400ms. Each tone lasted

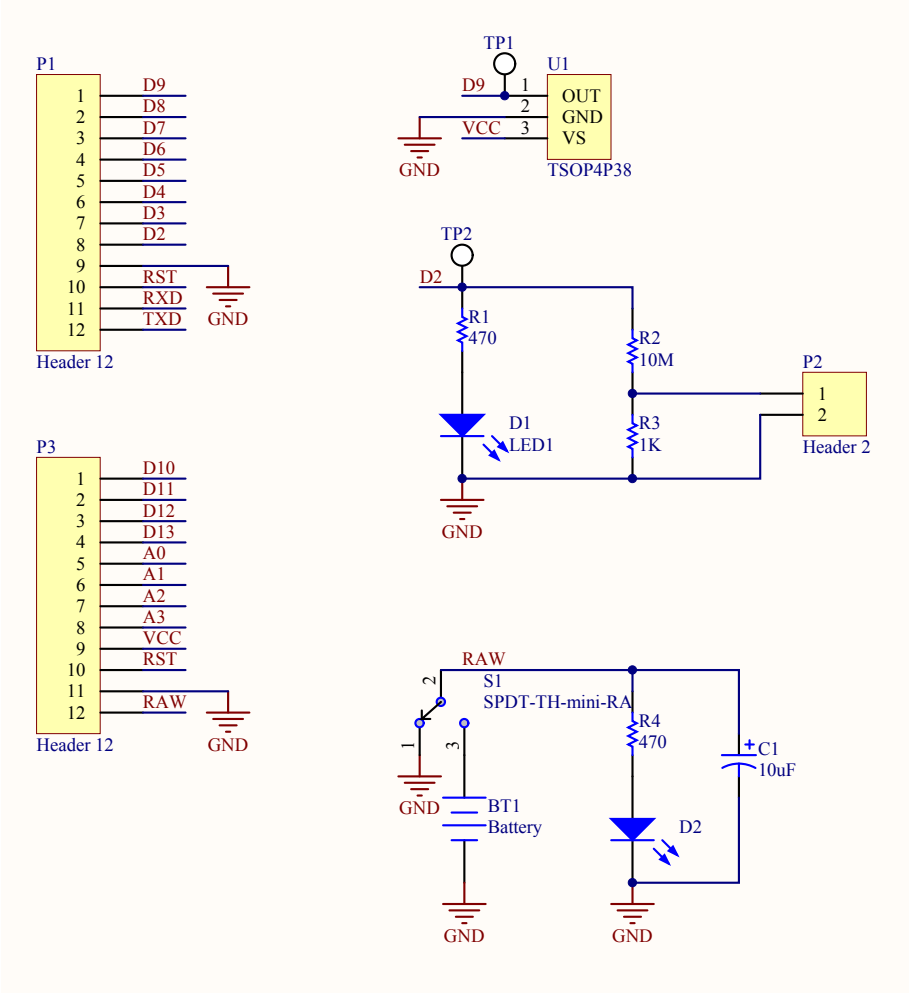

Fig. 3: Schematics of the receiver shield.

for 160ms. The computer's headphone socket was connected to the transmitter's audio socket (J2) using a 3.5mm to 3.5mm audio cable. The transmitter mode was switched to analog. The audio tone and the output pulse at the receiver (digital pin D2) were displayed on a digital scope (Tektronix MSO2012B) so that the delay between their onset could be measured.

F. Verification of the digital mode

Similar to Section II-E, this section describes the methods to verify the pulse width and the delay between the stimulus and the pulse. The phototransistor circuit described in Section II-A was connected to the digital input and the mode switch was set to digital. Two black-and-white checkerboard patterns were created as image files where one pattern is a reverse version of the other, following the guidelines in the ISCEV standard [9]. A Matlab code was written to display the patterns alternatingly every 400ms. The phototransistor signal sent to the digital

input D3 was displayed together with the output pulse at the receiver (digital pin D2) on the digital scope so that the delay between their onset could be measured.

III. Results and Discussion

A. *Results from the analog mode*

Figures 4 and 5 display the 1000Hz and 1200Hz audio tones with their corresponding pulses at the receiver unit respectively. The results show that 100ms-wide pulses were correctly produced when the 1000Hz tone was detected. On the other hand, 200ms-wide pulses were produced when the 1200Hz tone was detected. The delay between the onset of the 1000Hz tone and the onset of the pulse was 19.3 ± 0.1ms. The delay remained the same for the 1200Hz tone. The significance of the results is that correct pulse width was produced according to the audio tone and the variability of the delay was negligible. Hence the resulting evoked potentials can be shifted back by 19.3ms in order to compensate for the delay.

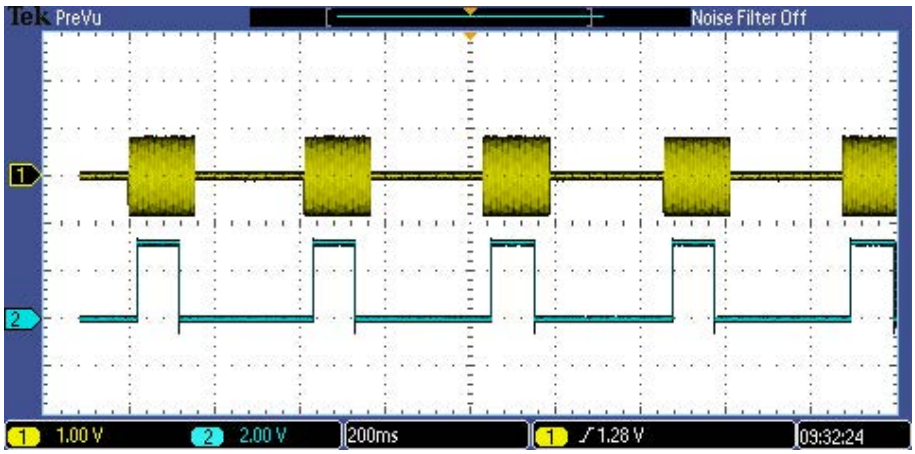

Fig. 4: Plot of the 1000Hz tones (top) and the corresponding 100ms-wide marker pulses (bottom) at the receiver unit.

B. *Results from the digital mode*

Figure 6 displays the output of the phototransistor circuit and the corresponding pulses at the receiver unit. The output of the phototransitor was high when the phototransistor detected light (i.e. a white check was presented in front of phototransistor) and low when the phototransistor detected no or very low light (a black check was presented). The results show that 100ms pulses were correctly produced whenever the phototransistor's output changed its state. The delay between the onset of the phototransistor's output and the onset of the pulse

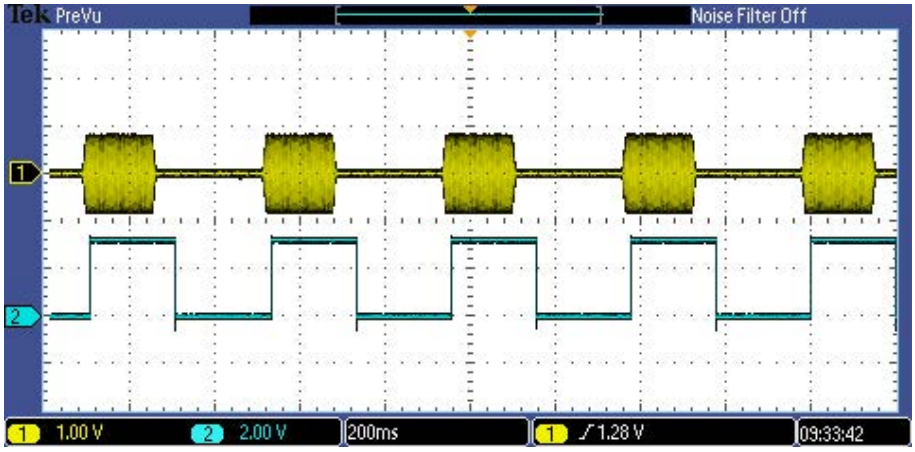

Fig. 5: Plot of the 1200Hz tones (top) and the corresponding 200ms-wide marker pulses (bottom) at the receiver unit.

was 6.3 ± 0.1ms. Similar to the analog mode, the variability of the delay was negligible. The delay was much smaller than in the analog mode since there was no buffering. The delay was dominated by the IR transmission.

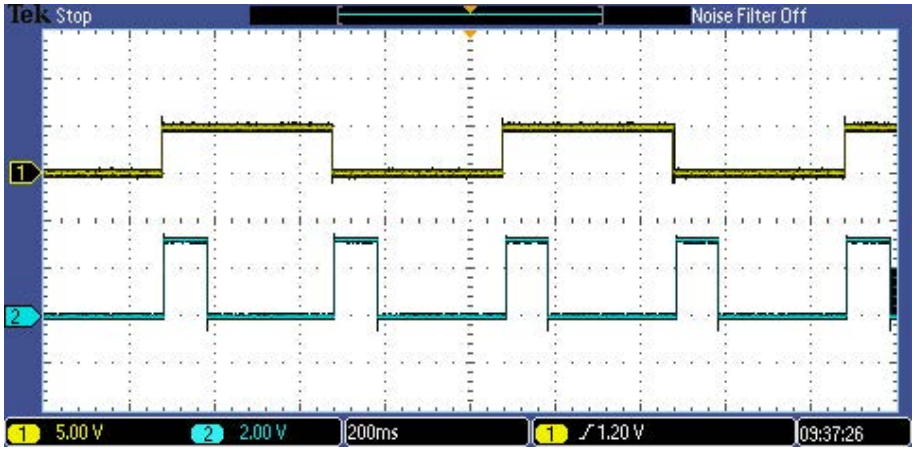

Fig. 6: Plot of the phototransistor circuit output (top) and the corresponding 100ms-wide marker pulses (bottom) at the receiver unit.

IV. Conclusions

The need for wirelessly transmitting stimulus markers has risen since the commercialisation of wireless EEG devices. As the devices are targeted to the consumer and developer market with a low cost, the system to transmit the stimulus markers should also be inexpensive, customisable and preferably open source.

This paper has described a wireless system that detects stimuli (audio or visual), transmits the information to the receiver unit which in turn embeds a pulse in the EEG data. The information of the stimuli is conveyed in the width of the pulse. The system has been verified for audio and visual evoked potential recordings. It can reliably produce pulses with correct widths according to the audio tones and whenever the visual stimulus reverses its pattern. Though a delay between the onset of the stimuli and the onset of the marker pulses is present, its variability is negligible. Hence the evaluated evoked potentials could be shifted back by the delay to compensate for the delay.

REFERENCES

[1] Y. Liu, X. Jiang, T. Cao, F. Wan, P. U. Mak, P.-I. Mak, and M. I. Vai, "Implementation of SSVEP based BCI with Emotiv EPOC," in *Proceedings of IEEE International Conference on Virtual Environments, Human-Computer Interfaces, and Measurement Systems,VECIMS*, Tianjin, 2012, pp. 34–37.

[2] S. Debener, F. Minow, R. Emkes, K. Gandras, and M. D. E. Vos, "How about taking a low-cost, small, and wireless EEG for a walk?" *Psychophysiology*, vol. 49, pp. 1449–1453, 2012.

[3] M. Duvinage, T. Castermans, T. Dutoit, M. Petieau, T. Hoellinger, C. Saedeleer, K. Seetharaman, and G. Cheron, "A P300-based quantitative comparison between the emotiv epoc headset and a medical EEG device," in *Proceedings of the 9th IASTED International Conference on Biomedical Engineering, BioMed 2012*, Innsbruck, 2012, pp. 37–42.

[4] A. Luo and T. J. Sullivan, "A user-friendly SSVEP-based brain-computer interface using a time-domain classifier." *Journal of neural engineering*, vol. 7, no. 2, p. 26010, Apr. 2010.

[5] Y. Maki, G. Sano, Y. Kobashi, T. Nakamura, M. Kanoh, and K. Yamada, "Estimating subjective assessments using a simple biosignal sensor," in *IEEE International Conference on Fuzzy Systems*, ser. 2012 IEEE International Conference on Fuzzy Systems, FUZZ 2012, Nagoya Institute of Technology, Japan, 2012.

[6] H. A. Lamti, M. M. Ben Khelifa, A. M. Alimi, and P. Gorce, "A brain eyes WHEELchair interface for severely disabled people assistance," REGIM: REsearch Group on Intelligent Machines, University of Sfax, National Engineering School of Sfax (ENIS), BP 1173, Sfax, 3038, Tunisia, pp. 686–694, 2011.

[7] N. A. Badcock, P. Mousikou, Y. Mahajan, P. D. Lissa, J. Thie, and G. Mcarthur, "Validation of the Emotiv EPOC EEG gaming system for measuring research quality auditory ERPs," *PeerJ*, vol. 38, no. 1, pp. 1–17, 2013.

[8] J. Thie, "A marker circuit to enable recording of auditory evoked potential using a wireless EEG system (EPOC) and a portable computer," *PeerJ PrePrints*, vol. 1, p. e2v1, 2013. http://dx.doi.org/10.7287/peerj.preprints.2v1

[9] J. V. Odom, M. Bach, M. Brigell, G. E. Holder, D. L. McCulloch, A. P. Tormene, and Vaegan, "ISCEV standard for clinical visual evoked potentials (2009 update)." *Documenta ophthalmologica. Advances in ophthalmology*, vol. 120, no. 1, pp. 111–9, Feb. 2010. http://www.ncbi.nlm.nih.gov/pubmed/19826847

APPENDIX A

TRANSMITTER CODE

```
/**************************************************
```

```
** Transmitter unit of the wireless marker system
************************************************/

#define set_bit(sfr, bit) (sfr |= (1 << bit))
#define clear_bit(sfr, bit) (sfr &= ~(1 << bit))

// Pin of the output pulse to the IR LED
#define outPulsePin   13

#define AUDIO_MODE_LEVEL      256
#define DIGITAL_MODE_LEVEL    512
#define SERIAL_MODE_LEVEL     768
#define INPUT_MODE_DEV        128

#define DIGITAL_INPUT_1       2
#define DIGITAL_INPUT_2       3
#define DIGITAL_INPUT_3       4
#define DIGITAL_INPUT_4       5
#define DIGITAL_INPUT_5       6
#define DIGITAL_INPUT_6       7
#define DIGITAL_INPUT_7       8
#define testPin               9

#define PULSE_WIDTH_1         50
#define PULSE_WIDTH_2         100
#define PULSE_WIDTH_3         150
#define PULSE_WIDTH_4         175
#define PULSE_WIDTH_5         200
#define PULSE_WIDTH_6         225
#define PULSE_WIDTH_7         250

// The IR emitter is operating at 38 kHz
#define IRFrequency           38000

// Settings for achieving a bit duration of 0.5 ms
// In practice the duration is set less than 500 us
#define IRDurationMicro      125
#define NUMCOUNT             40
```


PeerJ PrePrints | https://peerj.com/preprints/32v1/ | v1 received: 15 Jun 2013, published: 15 Jun 2013, doi: 10.7287/peerj.preprints.32v1

```
int outPulse = LOW;
volatile int toggleCount = 0;

// Declare global variables for the audio stimulus
// Analogue reference which corresponds to
// 2.5V (half of the power supply rail)
#define ANAREF              511
// Analogue threshold
#define ANATH               560
// Buffer size of the analogue samples
#define BUFSIZE             100
// Buffer to store the analogue samples
int audio_buffer[BUFSIZE];
// The threshold number of zero-crossings of the audio tones
// to differentiate between two tones
#define CROSS_THRESHOLD (13*BUFSIZE/100)

// Declare global variables for the visual stimulus
int vep_stim = LOW;

// Declare the variable type: InputMode and
// the functions associated with the variable type.
// If the function declaration is not included here,
// the compiler will return an error:
// InputMode does not name a type.
enum InputMode { AUDIO_INPUT, DIGITAL_INPUT, SERIAL_INPUT };
InputMode trigger_mode;
InputMode read_input_mode();

/*************************************************
** Initialisation function
*************************************************/
void setup()
{
  Serial.begin(9600);

  // Setup Timer2 which is used to create the 38 kHz pulse to drive the IR emitter
```


PeerJ PrePrints | https://peerj.com/preprints/32v1/ | v1 received: 15 Jun 2013, published: 15 Jun 2013, doi: 10.7287/peerj.preprints.32v1

```
  Setup_timer2();

  // Setup the inputs
  setup_input();
  setup_vep();
}

/*************************************************
** The main loop function
*************************************************/
void loop()
{
  // Determine the input mode using the analogue input A1
  trigger_mode = read_input_mode();

  // Determine the pulse_width according to the stimulus type
  byte pulse_width = 0;

  switch (trigger_mode)
  {
    case AUDIO_INPUT:
      pulse_width = run_audio_input_two_tone();
      break;
    case DIGITAL_INPUT:
      pulse_width = run_vep();
      break;
    case SERIAL_INPUT:
      pulse_width = run_serial_input();
      break;
  }

  if (pulse_width > 0)
  {
    Serial.println(pulse_width);
    // Write the pulse width to IR
    writeToIR(pulse_width);

    // Wait for a while
```


PeerJ PrePrints | https://peerj.com/preprints/32v1/ | v1 received: 15 Jun 2013, published: 15 Jun 2013, doi: 10.7287/peerj.preprints.32v1

```
    delay(500);
  }

}

/************************************************
** Generic initialisation of the pins and buffer
************************************************/
void setup_input()
{
  pinMode(DIGITAL_INPUT_1, INPUT);
  pinMode(DIGITAL_INPUT_2, INPUT);
  pinMode(DIGITAL_INPUT_3, INPUT);
  pinMode(DIGITAL_INPUT_4, INPUT);
  pinMode(DIGITAL_INPUT_5, INPUT);
  pinMode(DIGITAL_INPUT_6, INPUT);
  pinMode(DIGITAL_INPUT_7, INPUT);
  pinMode(testPin, OUTPUT);

  for (int n=0; n<BUFSIZE; ++n)
    audio_buffer[n] = 0;
}

/************************************************
** Specific initialisation of the pins for the visual stimulus
************************************************/
void setup_vep()
{
  pinMode(DIGITAL_INPUT_1, OUTPUT);
  pinMode(DIGITAL_INPUT_2, INPUT);
  digitalWrite(DIGITAL_INPUT_1, HIGH);
}

/************************************************
** writeToIREmitter sends a byte of data to the IR emitter.
** The LSB is sent out first.
** Bit 0 switches on the IR emitter while bit 1 switches it off. (active low)
** The duration of each bit is 3 ms.
```


PeerJ PrePrints | https://peerj.com/preprints/32v1/ | v1 received: 15 Jun 2013, published: 15 Jun 2013, doi: 10.7287/peerj.preprints.32v1

```
** The start and stop bits are 0.
**************************************************/
void writeToIR(byte data)
{
  // Write the start bit = 0 -> IR emitter ON
  // The ISR is called every 13 us.
  // If the duration is 3 ms and the signal period is 26 us, there must be 115 pulses.
  // Therefore the number of toggles is 115*2 = 230.
  IRon();

  // Write the data bits
  for (int i=0; i<8; ++i)
  {
    if ((data & 0x01) == true)
    {
      IRoff();
    }
    else
    {
      IRon();
    }
    data >>= 1;
  }
}

/**************************************************
** Switches on the IR emitter.
** The duration is determined by toggleCount.
**************************************************/
void IRon()
{
  toggleCount = NUMCOUNT;
  while(toggleCount > 0);
}

/**************************************************
** Switches off the IR emitter by setting toggleCount
** to zero and using delayMicroseconds(IRDurationMicro).
```

```
*************************************************/
void IRoff()
{
  toggleCount = 0;
  delayMicroseconds(IRDurationMicro);
}

/************************************************
** The main function for detecting the checkerboard visual stimulus.
** When the check reverses its pattern, it returns
** the specified pulse width.
** The function is blocking.
************************************************/
byte run_vep()
{
  while(vep_stim == digitalRead(DIGITAL_INPUT_2));    // Wait until the stimulus changes
  vep_stim ^= 0x1;                                    // Toggle the stimulus state
  return PULSE_WIDTH_2;
}

/************************************************
** The generic main function for detecting the digital inputs.
** The digital inputs are assumed to be active low.
************************************************/
byte run_digital_input()
{
  if (digitalRead(DIGITAL_INPUT_1) == LOW)
  {
    return PULSE_WIDTH_1;
  }
  else if (digitalRead(DIGITAL_INPUT_2) == LOW)
  {
    return PULSE_WIDTH_2;
  }
  else if (digitalRead(DIGITAL_INPUT_3) == LOW)
  {
    return PULSE_WIDTH_3;
  }
```


PeerJ PrePrints | https://peerj.com/preprints/32v1/ | v1 received: 15 Jun 2013, published: 15 Jun 2013, doi: 10.7287/peerj.preprints.32v1

```
  else if (digitalRead(DIGITAL_INPUT_4) == LOW)
  {
    return PULSE_WIDTH_4;
  }
  else if (digitalRead(DIGITAL_INPUT_5) == LOW)
  {
    return PULSE_WIDTH_5;
  }
  else if (digitalRead(DIGITAL_INPUT_6) == LOW)
  {
    return PULSE_WIDTH_6;
  }
  else if (digitalRead(DIGITAL_INPUT_7) == LOW)
  {
    return PULSE_WIDTH_7;
  }
  else
  {
    return 0;
  }
}

/************************************************
** The main function for detecting audio stimuli
** with only one tone
************************************************/
byte run_audio_input_one_tone()
{
  audio_buffer[0] = analogRead(A0);
  if (audio_buffer[0] > ANATH)
  {
    return PULSE_WIDTH_2;
  }
  else
  {
    return 0;
  }
}
```

```
/*************************************************
** The main function for detecting audio stimuli with two tones.
** When the tone is detected, the samples are stored in a buffer
** for counting the number of oscillations.
*************************************************/
byte run_audio_input_two_tone()
{
  // Store the most recent sample to the buffer
  audio_buffer[0] = analogRead(A0);
  // Check if the amplitude exceeds the threshold.
  // If so, an audio tone has been detected and it will start acquiring a frame
  // of samples for analysis.
  if (audio_buffer[0] > ANATH)
  {
    digitalWrite(testPin, HIGH);
    int num_cross = 0;
    // Fill the buffer with the samples
    for (int n=1; n<BUFSIZE; ++n)
    {
      audio_buffer[n] = analogRead(A0);
      // Analyse the samples to determine the type of tones
      // Evaluate the number of times the signal crosses the analogue reference
      // as an estimate to the frequency.
      // Only use positive-going edge.
      // Assume that the sampling rate is 10 ksamp/sec. Hence 1kHz tone should have
      // 10 positive-going edge.
      // In practice, 12 edges were returned for 1kHz tone and 14 edges for 1.2kHz.
      if ((audio_buffer[n] >= ANAREF) && (audio_buffer[n-1] < ANAREF))
      {
        num_cross++;
      }
    }
    digitalWrite(testPin, LOW);

    if (num_cross < CROSS_THRESHOLD)
    {
      return PULSE_WIDTH_2;
```


PeerJ PrePrints | https://peerj.com/preprints/32v1/ | v1 received: 15 Jun 2013, published: 15 Jun 2013, doi: 10.7287/peerj.preprints.32v1

```
    }
    else
    {
      return PULSE_WIDTH_5;
    }
  }
  return 0;
}

/*************************************************
** The main function for reading the pulse width
** from the serial input.
*************************************************/
byte run_serial_input()
{
  // Read data from the serial port if it is available
  if (Serial.available() > 0)
  {
    byte indata = Serial.read();
    return indata;
  }
  return 0;
}

/*************************************************
** Detect the input mode (audio, digital or serial)
** specified by the toggle switch.
*************************************************/
InputMode read_input_mode()
{
  // Read the raw input mode from the switch (analogue input A1)
  int inputModeRaw = analogRead(A1);
  if ( abs(inputModeRaw - AUDIO_MODE_LEVEL) < INPUT_MODE_DEV )
  {
    return AUDIO_INPUT;
  }
  else if ( abs(inputModeRaw - DIGITAL_MODE_LEVEL) < INPUT_MODE_DEV )
  {
```

```
    return DIGITAL_INPUT;
  }
  else if ( abs(inputModeRaw - SERIAL_MODE_LEVEL) < INPUT_MODE_DEV )
  {
    return SERIAL_INPUT;
  }
}

/*************************************************
** Definition of Setup_timer2() function
** The clock prescaler is set to 1.
** The waveform generation mode is set to CTC.
** According to the datasheet (page 148), fOC2A = fclk / (2*N*(1+OCR2A)
** where N is the prescaler and OCR2A the counter limit.
** The counter clock frequency is then 16MHz/1 = 16 MHz.
** The corresponding clock period is then 1/16 us.
** The desired signal frequency is 38 kHz and the corresponding period is 26 us.
** Then the toggle period needs to be 13 us which is the interval of the ISR call.
** Therefore, OCR2A must be 13/(1/16) - 1 = 207.
**
*************************************************/
void Setup_timer2()
{
  pinMode(outPulsePin, OUTPUT);

  outPulse = LOW;
  toggleCount = 0;

  // Set the prescaler to 8
  // Set CS22:0 = b010
  // Set the prescaler to 1 -> CSS2:0 = b001
  set_bit (TCCR2B, CS20);
  clear_bit (TCCR2B, CS21);
  clear_bit (TCCR2B, CS22);

  // CTC timer operation
  // Compare Output Mode = Toggle OC2A on Compare Match for non-PWM mode
  // Set COM2A1:0 = b01
```


PeerJ PrePrints | https://peerj.com/preprints/32v1/ | v1 received: 15 Jun 2013, published: 15 Jun 2013, doi: 10.7287/peerj.preprints.32v1

```
  set_bit (TCCR2A, COM2A0);
  clear_bit (TCCR2A, COM2A1);
  clear_bit (TCCR2A, COM2B0);
  clear_bit (TCCR2A, COM2B1);

  // Waveform Generation Mode = CTC
  // Set WGM22:0 = b010
  clear_bit (TCCR2A, WGM20);
  set_bit (TCCR2A, WGM21);
  clear_bit (TCCR2B, WGM22);

  // Set the counter limit
  OCR2A = 207;

  // Turn on timer2
  set_bit (TIMSK2, OCIE2A);
}

/*************************************************
** Definition of the Output Compare Match A Interrupt subroutine of Timer2
** The routine is called every 13 us given the timer configuration above.
*************************************************/
ISR(TIMER2_COMPA_vect)
{
  if (toggleCount > 0)
  {
    outPulse ^= HIGH;
    --toggleCount;
  }
  else
  {
    outPulse = HIGH;
  }
  digitalWrite(outPulsePin, outPulse);

}
```

APPENDIX B

RECEIVER CODE

```
/*************************************************
** Receiver unit of the wireless marker system
*************************************************/

#define IRFrequency          38000        // The IR emitter is operating at 38 kHz
#define IRDurationMicro      500          // bit duration of 0.5 ms

#define IRSensorPin          9
#define markerPin            2

int prev_sensor_output;

void setup()
{
  Serial.begin(115200);
  pinMode(IRSensorPin, INPUT);
  pinMode(markerPin, OUTPUT);
  prev_sensor_output = LOW;
}

void loop()
{
  // Check if the transmitter sends data
  byte pulse_width = 0;
  int sensor_output = digitalRead(IRSensorPin);
  if ((sensor_output == LOW) && (prev_sensor_output == HIGH))    // A start bit is detected
  {
    // If the start bit is detected, read the data
    pulse_width = IRread();
    Serial.println(pulse_width);
    // Send the marker signal
    digitalWrite(markerPin, HIGH);
    delay(pulse_width);
    digitalWrite(markerPin, LOW);
```


PeerJ PrePrints | https://peerj.com/preprints/32v1/ | v1 received: 15 Jun 2013, published: 15 Jun 2013, doi: 10.7287/peerj.preprints.32v1

```
  }
  // Keep track of the sensor output
  prev_sensor_output = sensor_output;
}

/*************************************************
** Transmitter unit of the wireless marker system
** IRread detects the state of the IR sensor and interprets it to bits.
** After reading 8 bits and detecting the stop bit, the data is returned.
*************************************************/
byte IRread()
{
  // Skip the start bit and jump to the middle of the first bit
  delayMicroseconds(IRDurationMicro*1000*1.5);
  // Start reading the 8 bits
  byte data = 0;
  byte mask = 1;
  for (int i=0; i<8; ++i)
  {
    int inbit = digitalRead(IRSensorPin);
    if (inbit == HIGH)
    {
      data |= mask;
    }
    mask <<= 1;
    delayMicroseconds(IRDurationMicro);
  }
  return data;
}
```


PeerJ PrePrints | https://peerj.com/preprints/32v1/ | v1 received: 15 Jun 2013, published: 15 Jun 2013, doi: 10.7287/peerj.preprints.32v1